Master Thesis

Adaptive Support of Knowledge Work by Analysis of User Objectives

by Matthäus Martynus

Academic Supervisor: Prof. Johannes Fürnkranz
Knowledge Engineering Group
Darmstadt University of Technology (TUD)
– Computer Science Department –

Industrial Partner: SAP Research Darmstadt
Supervisor: Robert Lokaiczyk

Darmstadt, September 2008

Contents

List of Figures

Chapter 1

Introduction

This master thesis investigates how the analysis of user objectives can can be used to give an adaptive support to a knowledge worker during his daily work. The thesis is hosted by the Research Department of SAP at its Campus Based Engineering Center in Darmstadt.

1.1 Motivation

During the daily knowledge work it is often the case that we are recurring things. We again and again click us through the folders to find a file because we do not want another shortcut on the desktop although we use this file very often, especially when we also working on the one file we already have opened. And there are several other resources that we often use together with this two files. It would be very helpful to have an assisting system that would facilitate the finding and using of this resources.

Another aspect is that during our work we often come to a point when we need some information about a particular thing. In this moment we are switching from a worker to a learner. During todays work we can not use anymore the old approach *learn first, apply later*. Today the learning and applying is mixed up. But before we can start to learn we first must find the resources from which we can learn. It is not possible to release us from the learning, but it would be be really nice when something could support us by finding for us the resources from which we can learn.

Everything we are doing could be classified in tasks. Every task is associated with resources that we need to use to fulfill the task. For some of this resources we know where we can find them, for some we do not. And some others we do not even know that we have to use them. Particularly

when we are doing this task for the first time. It would save us a lot of time and even preserve us from forgetting something important when a system could realize on which task we are working at the moment and give us a list of resources that are needed to finish this task correctly.

Surely, there are many systems that can help us. We can create shortcuts or favorites to save the time during browsing. We can use static support system to get help about how to use an application. We can search the internet to find the information we need. But all these steps require an initiative from us. The main idea of this thesis is to investigate how a system that is analysing our working environment can find out what our current objectives are and give us automatically a dynamic support that is adapting to our needs.

1.2 Practical Example

With this short practical example we would like to introduce how how the support of a knowledge worker could look like. We will show how we can support three user goal types during his work.

Let us assume that the knowledge worker John is currently working on a project: He is implementing an application and writing also a documentation for it. For this reason he has to open several files and browse through several folders. From the past our system learned which files and folders often have been opened in temporal coherence. The correlation of the opened resources in the past and the resources that are opened at the moment are the basis for the recommendation of resources to John. For example as soon as he opens the documentation file our system should provide him with a list the among others containing the project file which can be opened by one click. Additionally the system learned that the knowledge worker often visits the website of this project during working on it. So it also recommending a link to this website in the list of resources. This is support of the **Navigational Goal**.

During his work John gets an email from another person also working on this project. This person requests to implement a special algorithm in this application. The email contains a short description of this algorithm, but John does not completely understand the description. Normally he would use the web search or some company document repository searches to find some more detailed descriptions of this algorithm. Our system is able to extract from the email the name of the algorithm: It appeared in the subject and several times in the body of the email. With this name

and maybe some other correlated terms our system searches the web and the company document repository for useful resources that contain some detailed information about the algorithm. The search is done completely in the background and John gets a list of the found useful resources which he can open by a single click. This is support of the short term **Informational Goal**.

All the knowledge work can be separated smaller parts. Each of them has to to do with concrete task. For example the actual task of John is *working on the project*. Beside the things that we described above this task also requires maintaining of working time document and sending a daily update by email to the project lead. Maybe John is new on this project and he does not know that he has to do this. Our system has monitored his work and has classified it to the task *working on the project*. By classifying it with this task it recommends a list of resources that has been previously assigned to this task. This gives John a hint what he still has to do and he again can use the recommended resources by simply clicking on them. This is support of the **Transactional Goal**.

As we described our system **analyses** the work of John by monitoring it and extracting information on what he is currently doing. This information is being used to give him an **adaptive support** that tries to provide him with resources that can useful for him.

1.3 Structure of the Thesis

In Chapter 2 we will explain why we choose the Informational, the Navigational and the Transactional Goal as the user objectives we want to analyse and support. First we characterize the three goals in Section 2.2, 2.1, and 2.3. After that Section 2.4 provides information about the analysis we are performing before Section 2.5 describes how our adaptive support distinguishes himself from other support systems.

Chapter 3 introduces the implemented system in all his parts. Section 3.1 describes the context monitor which collects the information about the users working environment. An overview over the the parts of our system that are located in the database is given in Section 3.2. Section 3.3 presents the user interface that makes the recommended resources available user. It also has a part that was implemented specially for the evaluation. Section 3.4 describes a tool tool that was also implemented to emulate what resources the users would have seen during their work when our system would have run.

Chapter 4 consists of a description of the implemented algorithms. Section 4.1) describes how extract the navigation resources and algorithms that use them to provide the user with other navigation resources and support the Navigational Goal. The algorithms supporting the Informational Goal (Section 4.2) are based on methods from the natural language processing area. The algorithms that support the Transactional Goal are introduced in Section 4.3 and are all classifiers which try to classify the actual work of the user.

Chapter 2

User Goals

The main theme of this Master Thesis are the user goals that the knowledge worker wants to accomplish. The are several categorizations for the user goals in this field of study. Following [JBS07] we will distinguish three of them: The *Informational Goal*, the *Navigational Goal*, and the *Transactional Goal*. This three categories or variations of them have been used in several studies. [JBS07] affirms approximately 75 percent of web search queries can be classified in one of this three categories. This estimation can be transfered on the knowledge work on the desktop since this is the location from which the web search queries are started.

In this chapter we will describe the three categories in detail and the *Adaptive Support* based on the *Analysis* of the working environment of the knowledge worker.

2.1 Navigational Goal

In many cases the knowledge worker has to work on several files or folders, visit some websites and communicate with some people to fulfill a task. All this are navigational objects and the *Navigational Goal* of the knowledge worker is find them on the local hard drive or the internet. When he or an another person is working on the same task, then in is probable that he will use similar navigational objects or sequences of them like like he or other persons have used in the past. Sometimes the location (the local path, the URL or the name of a person) is unknown so that the knowledge worker has to search for it. The search of the web or the browsing through local folders could be eliminated when based on the last used navigational objects and the previous history the program provides the needed objects and the user

needs only one click to use them.

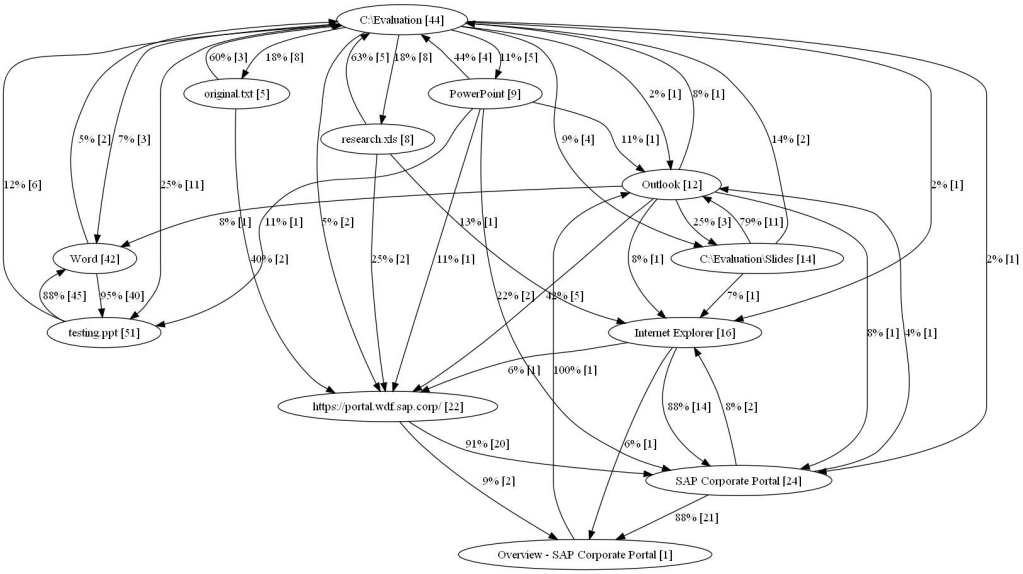


Figure 2.1: Navigation Graph

We can create a navigation graph like in Figure 2.1 from the navigation objects. Each navigation object is represented by a node with edges to all navigation objects that occurred as a successor of it in the history. On this graph we can use partitioning algorithms and propose all navigation objects in the actual partition that have not been accessed in the actual session. An alternative could be a searching for sequences matching the last few navigation objects and proposing the navigation object that have followed the found sequences.

2.2 Informational Goal

When a knowledge worker is searching for some information then his main purpose is an *Informational Goal*. The kind of information can be provided by several types of resources. Following [LGFM08] we selected the following five information resources:

- *Definition* - a general description of something. It provides the information in a short descriptive way. One example for this is the abstract section of the Wikipedia[1] articles. This kind of information is mostly recommended for advanced users since the understanding a definition often requires earlier knowledge.

[1] http://www.wikipedia.com

- *Example* - a concrete usage of something. This could be websites, documents or pictures containing the needed information. One source for this information can be the DBpedia[2] project, which is a collection of semantic relations between objects which are extracted from the Wikipedia metadata. This information kind can also be used by unexperienced users, since the understanding of a concrete usage requires less intellectual ability of abstraction.

- *Essay* - an extensive document that provides the information with the needed background. This could be papers, articles or presentations that cover the information. It could be difficult to find an essay containing the needed information and to extract this information from there. One source for this kind of information could be document repositories of a company that contain the domain-specific knowledge.

- *Question and Answer* - the request for a concrete information with the associated help from others. The easiest way to get an information is to ask for it. Many questions has already been asked and most of them also have been answered. One collection of this kind of information is the WikiAnswers[3] website. This information can be highly useful, but it could be hard to find the question whose answer provides the user with the needed information.

- *Instruction / User Manual* - a step by step description of something. When starting to work on something the first time some users prefer a step by step description of it. This information often is provided by user manuals contributed with programs. One website that collects information in this form is WikiHow[4]. This kind of information is most useful when the user is at the beginning of a process that requires several steps for completion and he does not know which steps have to be done in which order.

To find out which information the knowledge worker is searching for we can analyse the current context and extract terms from the typed terms, clipboard contents, window titles, emails, files and websites. Since there are many useless terms under them it is an option to use natural language processing techniques to filter them and to select the most important ones. This selected terms can then be used to choose documents from a database or to use a search and propose the results to the knowledge worker.

[2] http://www.dbpedia.org
[3] http://www.wikianswers.com
[4] http://www.wikihow.com

2.3 Transactional Goal

The support of the *Transactional Goal* of a knowledge worker is a little bit different to the two described before. It is based on fact the knowledge worker is recurrently working on the same tasks. The support works in another dimension then the support for the other two goals. It is not a direct support that provides resources depending directly on the collected context information. The program uses the context information to identify the actual task and then proposes resources to user that have been previously assigned to the task. This resources could be applications, contact data of experts for this task, local files needed to fulfill this task or websites which must be visited.

One approach to identify the actual task is to train a machine learning algorithm with a prelabeled training data. The training data would be the collected context information. When the identification of the actual task is correct, then provided resources are surely useful for the knowledge worker. In contrast to this a false classification results in resources that have nothing to do with the actual task.

2.4 Analysis

To support a knowledge worker during his work we have to analyse his goals. In this thesis we concentrate our *Analysis* on the computer desktop as our context. We try to collect as much information as possible by monitoring the user actions and the operating system. The collected information must be filtered and analysed. For each of the three user goals the useful information can be different.

It is very hard to distinguish which of the three previously described goals is the most important one for a user at a particular time. So we decided to support all three goals at anytime. Several analysing algorithms can run simultaneously and all of them will be provided with all collected information.

2.5 Adaptive Support

A knowledge worker can get many static support from the help of an application or from some web sites or from a co-worker. The main point in this support is that it must be actively requested and is not automatically

adapting to the knowledge worker's goals. Concrete terms must be typed in a search or be described to the other person to get the needed information.

In contrast to that the *Adaptive Support* analyses the knowledge worker's context in the background a proposes automatically resources which could be useful. This resources are presented in a discreet way so that they are not disturbing the user but always accessible. If he needs some support he can utilise them directly without requesting them afore.

To describe how our adaptive support differs from other support systems we classify in in four of the dimensions described in chapter 11 of [Her94]:

- *Initiative for activating the support system*

  A support system can be either *passive* or *active*: Most of the existing support systems are passive. The user must explicitly start them to get some support. Some others are active ones. They open a dialog or pop up without an explicit request for it from the user.

  Our support is passive: Although it is updating his contents actively it does not disturb the user during his work. When the user needs support he is free to click on one of the provided resources.

- *Context aspect of supporting information*

  The provided supporting information can be either *static* or *dynamic*: A static supporting information provides the same resources every time it is requested. The resources are statically linked to the terms the user typed or clicked. Our support system provides the information dynamically. It collects the context information of all users and learns from the collected date from the past which resources users have used depending on their context. The provided resources are depending on the actual context of the user.

- *Individuality of supporting information*

  Most support systems provided *uniformed* information for all users. That means that each user will get the same resources when he makes the same request or has the same context status. Our support is *individual*: Depending on the information how the user used it in the past the provided resources can adapt to the preferences of the user.

- *Integration of the support system*

  Normally a support system is *integrated* in an application. It can provide only information for this application. Our support system is *application independent*. It provides support at any time for each task.

The user do not have to become comfortable with several support systems, but can use only one to get the needed information.

Summarising it can be argued that our *application independent* system provides a *passive* support that is *dynamically* adapting to the users context and gives him *individual* ressources which he maybe could need.

Chapter 3

Implementation

The application that was implemented for this thesis can be separated in four parts: The *Context Monitor*, which is running in the background and monitoring the user actions during the knowledge work, the *Database*, which is storing the collected information and also the configurations for the user interface and the algorithms, the *SAPSidebar*, which is the user interface and contains the implemented algorithms for the recommendations, and the *User Interaction Emulator*, which emulates the Context Monitor and by this the interaction of a user.

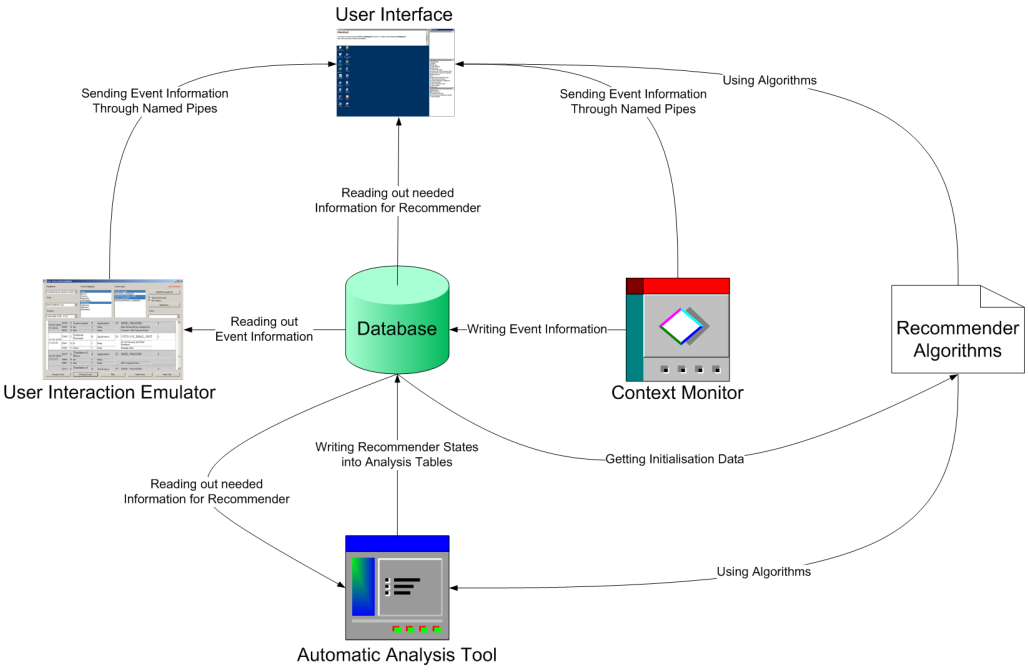


Figure 3.1: Communication Diagram

The communication between these parts is described in Figure 3.1. All

three software parts are communicating with the database. The Context Monitor is sending the data of the collected user interaction events to the database. The User Interaction Emulator is reading out the event data from the database to emulate the user interaction for the SAPSidebar. The SAPSidebar gets their configuration from the database and the algorithms request the information they need. The three software parts do not know from each other. They all reference the *Events.dll* which provides the needed interfaces and objects for the communication through *Named Pipes* [Coo08]. This way either the Context Monitor or the User Interaction Emulator can feed the SAPSidebar with events and there is no difference for it where the events are coming from.

3.1 Context Monitor

The Context Monitor has an amount of monitoring processes which catch events occurring on the computer during the user's work. Figure 3.2 is showing the collected event categories and types that are used in the implemented algorithms. They are described below. A more detailed and structured description can be found in the appendix. Each event stores the category, the type, and the time when an event occurred. The specific information are stored as event attributes of the events.

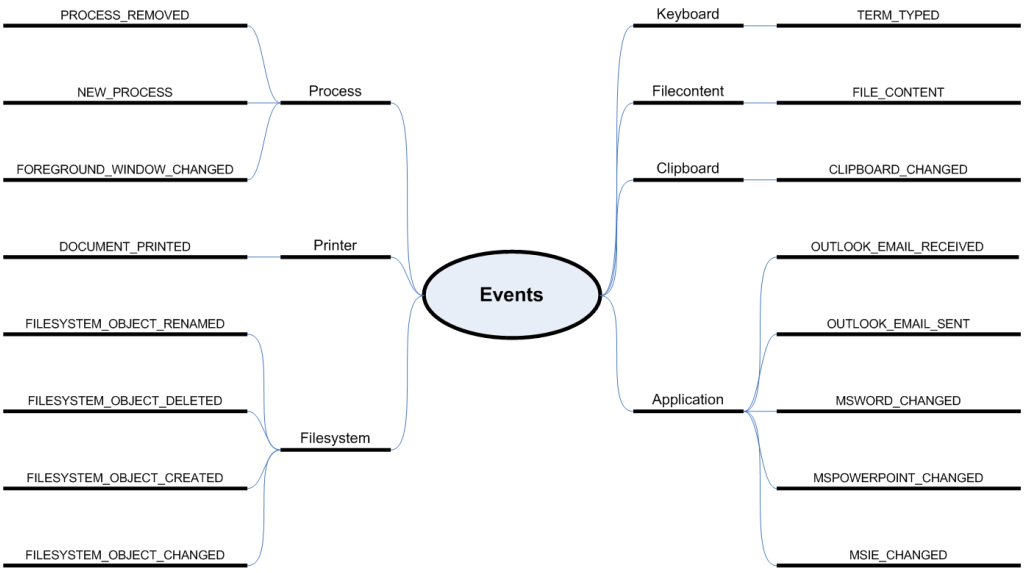


Figure 3.2: Event Categories (Level 1) and Types (Level 2)

The Context Monitor can store the events in a local file or send them to the database. It has also a word segmenter that extracts words (tokens) from a given text.

3.1.1 Keyboard Events

There is one keyboard event that stores terms typed in by the user. The input stream of pressed keys is analysed to get the terms. The typed characters are collected until a special character, like for example a whitespace, occurs which ends the actual term. A press on the backspace key reduces the length of the memorized term by one.

3.1.2 Clipboard Events

The only one clipboard event catches changes of the clipboard content. If the content is a text then it is stored as one event attribute. Beside this the word segmenter is used to extract all occurring words that are also stored as event attributes.

3.1.3 Filesystem Events

There are four file system events: They occur when a file system object (file or folder) is created, changed, renamed, or deleted. The events store the full path of the object (two paths when it is a rename event) the extracted words by the word segmenter as event attributes. Additionally an attribute stores the type of changing: At the creation, renaming, and deleting events it only indicates the object type, at the change event it also describes the attribute that has changed.

3.1.4 Filecontent Events

For each change of the *LastAccess* attribute of a file the filecontent event saves the file name and the words that occurred in the file content as event attributes.

3.1.5 Process Events

One type of the process events are changing foreground windows. These events occur each time when the name of the top most window changes. As event attributes the window title and the associated process name are stored. Beside this the word segmenter is used to extract all occurring words from the window title and to store them as additional event attributes.

The other type of process events occur when a process is started or removed. The list of running processes is monitored and every change results in a new event. The process id and the process name are added as event attributes.

3.1.6 Application Events

The application events are created by monitoring some specific applications. At the time when this thesis was written the following four applications have had their special monitors:

Microsoft Outlook

Currently the Microsoft Outlook monitor catches two events: When the user receives or sends an email. In both cases the email subject and the email address of the sender respectively the receiver are stored as event attributes. Additionally the word segmenter is used to extract words from the email subject and body, which are also stored as event attributes.

Microsoft PowerPoint and Microsoft Word

For these two applications an event is created when something changes within them. In such cases the title and the words extracted from the document name and content by the word extractor event attributes of this event. In addition in Microsoft Word the words of the actual selection are extracted and added to the event as event attributes.

Microsoft Internet Explorer

The monitor of the Microsoft Internet Explorer also creates an event when something changes within the application. The name and URL of the actual website and the words extracted from the name, the actual selection, and the site content are the event attributes. Before the content is submitted to the word segmenter the HTML tags are filtered.

3.1.7 Printer Events

Every time when a document is printed a printer event is created. As event attributes the document name, the owner of the document, the printer on

which the document is printed, and the words extracted from the document name are stored. The printer events are created for all documents printed by all users, but they can be filtered through the owner event attribute.

3.1.8 Word Segmenter

The word segmenter replaces in the given text all characters which are not alphabetic or a hyphen with whitespaces. After that all consecutive whitespaces are replaced by one whitespace. The resulting text is splitted at the whitespaces. All single character tokens are ignored, the others are processed and a set with found words and all their positions is returned.

3.1.9 Storing of Event Data in Database

The writing of the event data to the database is done at one central place. Three stored procedures are called to send the data of the event (I), the data of the event attributes (II), and the positions of each event attribute (III). The using of stored procedures makes the communication between Context Monitor and database more secure and more efficient, but the most important advantage is that the database-side processing of the send data can be changed without changing the Context Monitor software. The only requirement is that procedure signature must stay the same.

3.1.10 Named Pipes Communication

Named pipes are part of the .Net Framework and allow processes that are running on the same machine to communicate with each other. This processes does not have to know each other, they only have to share some interface declarations for the local communication channel. This technology makes it possible to replace the Context Monitor with the User Interaction Emulator without changing something at the user interface. Even the configuration does not have to be adjusted.

3.2 Database

The used database is a MySQL 5.1 database [AB08]. It has three main parts: It is the storage for the collected event data, it contains the configuration

for the user interface and the implemented algorithms and a few algorithms have their main part in tables and stored procedures in the database.

3.2.1 Storage

The main tables of the event storage part in the database are:

- The table *monUser* stores the user specific information.
- The table *monEvents* stores the events with ids of user, event category and event type.
- The table *monEventAttributes* stores the events attributes with ids of event, attribute name and attribute type. It has one value field which either an integer value (at numeric attributes) or the id of a string value (at text attributes) which is stored in the following table.
- The table *monEventAttributeStringValues* stores all strings the occurred in one of the attributes.
- The table *monEventAttributePositions* stores the positions on which an attribute occurs in a context. Currently it used to store the positions of words extracted by the word segmenter in the initial text.

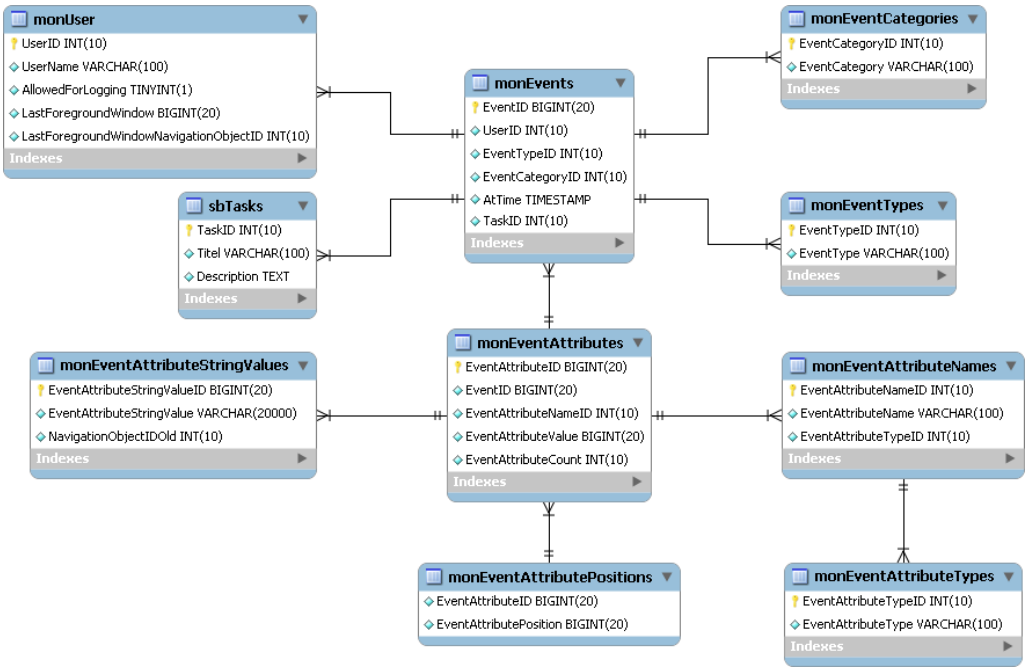


Figure 3.3: Database Structure

Their associations through foreign keys can be seen in Figure 3.3. The other tables are needed to define the ids of the event categories, event types, attribute names, attribute types, and tasks.

3.2.2 Configuration

There are two tables that contain all configuration for the user interface and the algorithms that are used to fill the recommendation lists. The table *sbUserSettings* is connected per id to the user table described above. It has one field that determines what type of user interface (either the sidebar or the topbar) should be started. Another field contains the id of the configuration that should be used for the user interface. When the sidebar is used then there can be multiple entries in the table *sbConfigurations* with the same configuration id. Each entry contains the settings for one recommendation algorithm that fills one list in the sidebar.

3.2.3 Algorithms

A few of the algorithms has some logic and precomputed data in the database. The logic is located in stored procedures. As long as a sequential computation of values is not necessary it is much more efficient to do the computation in the database and return the computed results instead of loading the complete data from the database and doing the computation in the software.

During the writing of the event data a precomputation for some algorithms is done. The big benefit is that the data is computed only one time and stored in tables in contrast to a recurrently computation every time when an algorithm makes his recommendations.

3.3 User Interface

The User Interface has two forms: The *sidebar*, which can display multiple recommendation lists, and the *topbar*, which was designed for the field study and has a task description region and an optional recommendation list.

The UML class diagram [Fow97] in Figure 3.4 the classes of the SAPSidebar and their relations. The most important classes are described below. A complete and formal description of the classes can be found in the appendix.

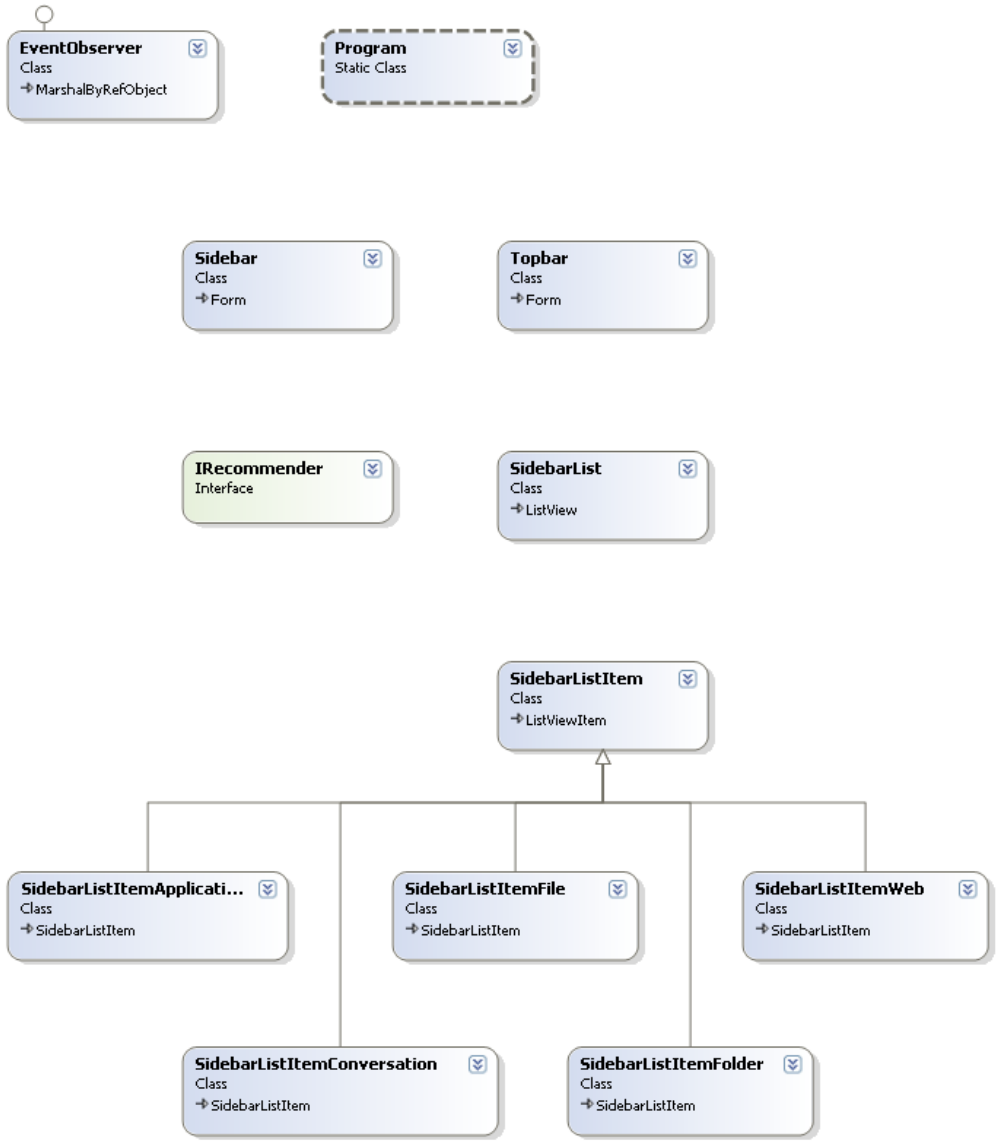


Figure 3.4: UML Class Diagram of the User Interface (*noch nicht fertig*)

- The class *Program* contains the static method that is executed at program start. In this method the EventObserver is initialized, the configuration loaded from the database and either the Sidebar or the Topbar started.
- The class *EventObserver* is a wrapper class for the communication through Named Pipes. The events send from the Context Monitor or the User Interaction Emulator arrive at the EventObserver which forwards them to the Sidebar or the Topbar, who forward them to their recommender algorithms.
- The classes *Sidebar* and *Topbar* are the two forms of the user interface. They are described in detail below.
- The interface *IRecommender* defines the methods which each recommender algorithm must implement. The recommender algorithms have an assigned SidebarList, which is filled by them, and receive all events. The filter them by type and use only the needed ones.
- The class *SidebarList* is derived from the class *ListView*. It contains SidebarListItems which are displayed with their assigned icon in front of the item text.
- The class *SidebarListItem* is derived from the class *ListViewItem*. It is extended by the functionality that a click on the item in the list fires an action which is different for each of the five recommendation object types, which are all represented as a class derived from the SidebarListItem:
  - The *Application* item represents a software that is started when the user clicks on the item. When it is possible the icon before the item is typical for the software.
  - The *Conversation* item represents a communication with another person. Currently an email to the person is opened when the item is clicked. The text of the item is the name of the person.
  - The *File* represents a file on the local hard drives. After a click on this item it is first checked if the file exists and when it is so the file opened by the assigned default application. The displayed icons are set by the default application assigned to the file ending.
  - The *Folder* item represents a folder on the local hard drives. A click results in an existence check and an opening of the folder in the explorer when it exists.

– The *Web* item represents a web location specified by their URL. The name of the item is the document name displayed the last time when this URL was loaded. A click on this item opens the URL in the default web browser.

3.3.1 Sidebar

The Sidebar is docking to the borders and reserving the place so it is always visible. The initial position is on the right side of the desktop. Depending on the count of recommender settings for the selected configuration the same amount of SidebarLists is created. Each of the lists is assigned to one of the recommenders that are configured by the loaded settings entry. All lists have the same size and are automatically arranged in one row or one column depending on the form of the Sidebar.

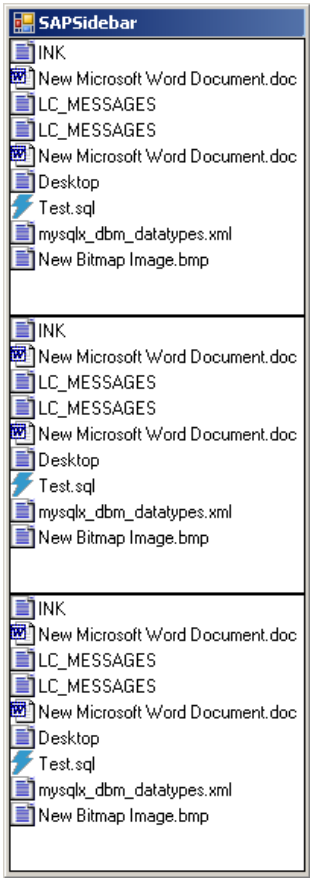


Figure 3.5: SAPSidebar GUI - Sidebar

This implementation is perfect for the manual analysis phase. The amount and the configurations of the SidebarLists can be changed in the database. The used configurations can remain in the database and can be

reassigned at a later time by simply setting the `ConfigurationID` for the user.

3.3.2 Topbar

The Topbar is docking in the same manner like the Sidebar, but the initial position is on the top. It has a rich-text-format textfield in which the tasks are displayed. They are loaded in a random order at startup. At the beginning the guidelines are displayed. There is one button which has to be clicked to get the first task, to switch to the next task after finishing the actually displayed one, and to finish the evaluation after the last task. The one SidebarList is optional. It is only displayed and filled when a configuration is assigned. From this configuration only the first entry is used because the Topbar can not have more than one list.

Figure 3.6: SAPSidebar GUI - Topbar

In the first phase of the evaluation the Topbar is used to guide the work the test persons are doing. They have no recommendation list and have to accomplish the tasks without any support. The Topbar notifies the Context Monitor when a task switch occurs so that the collected event data is labeled with the corresponding task ids.

3.4 User Interaction Emulator

The User Interaction Emulator emulates a user working on the machine by replacing the Context Monitor and sending the events through the Named Pipes. First the user whose interaction should be emulated must be selected. Then the event categories and types have to be chosen to restrict the amount of events. After selecting the session the events can be loaded from the database. Alternatively the events can restricted by a SQL statement which returns the `EventID`s which should be loaded from the database.

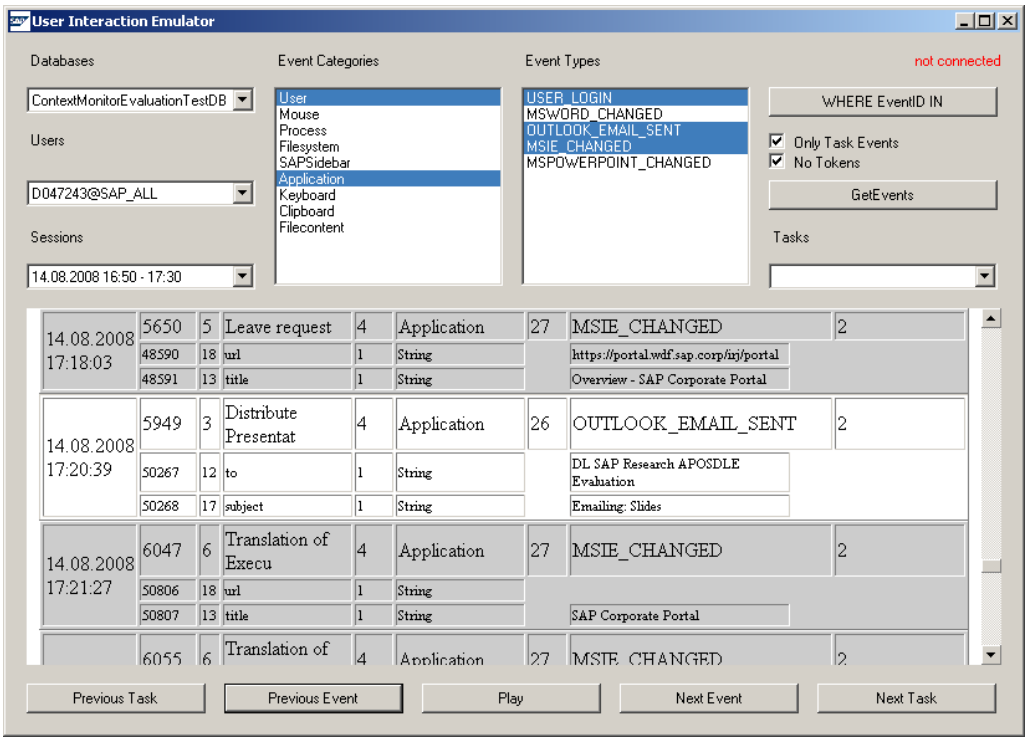


Figure 3.7: User Interaction Emulator

The events are displayed in the browser control. The actual event has a white background, all others a gray one. The events can be send manually by going one event forward or backward. Beside this it is possible to jump to the beginning of the previous or the next task. Alternatively the task can be selected from the combobox above the display. It is also possible to let the User Interaction Emulator send the Events in real time based on the time when they have been collected.

Chapter 4

Implemented Algorithms

The implemented algorithms are all based on the events which have been collected in the past and which are actually collected and send to them. The events provide the algorithms with information about the user interaction and his current context. This information is used to compute resources which are probable useful for the user in his current situation.

The algorithms are separated by the user goal which they are supporting. Each of the three information goals has his own type of algorithms which uses a subset of the events.

4.1 Navigational Goal

All algorithms supporting the navigational goal are based on the same data: The use foreground window change events to extract the navigation objects which represent resources the user is working with. This navigation objects are the input data for the algorithms and represent also the resources recommended to the user.

4.1.1 Navigation Objects

A foreground window change event only gives us the title of the actual foreground window. This title mostly contains the name of the application to which it belongs and some information about the content of this window. The navigation objects can be classified by the extraction strategy that is used to extract them from the window title. Most of them use some information of other events that occurred right before the foreground window change event. This extraction is done in the database at the moment when

the events are written to it. There are six different extraction strategies:

- *Complete file path in title* - in such cases the name of the file (the part after the last slash) is used as the name of the navigation object and complete path as its target.
- *File name in title* - The file name is used as the name of the navigation object. But it can not be used directly as its target since it is not known in which folder this file is. It would be very inefficient and probably ambiguous to search for this filename on the local hard drives. Instead we are searching the most recent filesystem event that was associated with a file with this name. This event provides us with the complete path which is used as the target of this navigation object.
- *Message* - this term with a hyphen before it indicates an opened email in Microsoft Outlook. The text before the hyphen is the subject of the email. We are using the events for received and sent emails and get the sender respectively the recipient of this email. With this information we create an navigation object that has the name of this person as name and the opening of an email to this person as target.
- *Microsoft Internet Explorer* - the title of a window of the Microsoft Internet Explorer contains the title of the actual website. This title is used as name of the navigation object and to search the last IE change event that affects the opening of a website with this title. This event then provides us with the address o this website which is used as target of the navigation object.
- *Folder* - The explorer has always the complete path of the actually opened folder as title. It can be directly used as target for the navigation object which opens the folder when it is selected. The name of the navigation object is the name of the folder (the part after the last slash).
- *Application* - When an application is started without opening a file it has only their name in the title. In such cases we create a navigation object that starts this application when it is selected. As target it has the path of the executable and as name the term `Start` concatenated with the application name.

The navigation objects are stored in the database. With each of them *supporting users count* is stored that represents the amount of users that

has already used this navigation object. The algorithms select the data on which their recommendations are based by this number to filter navigation objects that have been used by only one or a few users.

The extracted navigation objects are assigned to the events. To make the further computation more efficient the ids of the events with an assigned navigation object are stored together with the id of the preceding foreground window event with an extracted navigation object. Based on this data a sequence of the used navigation objects can be concatenated.

4.1.2 Recent Objects Recommender

The *Recent Objects Recommender* simply fills the recommendation list with the last extracted navigation objects. The most recent one is always the first in the list. This dummy recommender is based on the circumstance that the users sometimes use resources that they have used some time before. We have implemented this recommender to have a reference value for the accuracies of the other algorithms.

4.1.3 Sequence Probability Recommender

The *Sequence Probability Recommender* stores the last N navigation object. Then the sequences in the database searches for subsequence that exactly match this N navigation objects. The recommendation are the navigation objects that have followed this sequence ordered descending by the number how often this sequence with the length $N + 1$ have been found. The value by which the recommended navigation objects are ordered is the probability that the recommended navigation object will follow the sequence:

$$P(O) = P(O \mid \langle X_{-N}, \ldots, X_{-1} \rangle) = \frac{n(\langle X_{-N}, \ldots, X_{-1}, O \rangle)}{n(\langle X_{-N}, \ldots, X_{-1} \rangle)} \tag{4.1}$$

This recommender has his entire logic in stored procedures in the database.

4.1.4 Graph Probability Recommender

The *Graph Probability Recommender* computes a directed weighted graph based on subsequences with the length 2. The nodes represent the navigation objects. The edges represent that one navigation object occurred after

an other while the weight of an edge represents the number how often that happens. The algorithm either selects the last N navigation objects or the navigation objects that occurred in the last T seconds. All outgoing edges of the associated nodes are grouped by their destination nodes. The associated navigation objects are recommended and ordered descending by the sums of the weights of the selected edges. The sorting value is proportional to the probability that that a navigation object will occur given that one of the select navigation objects occurred:

$$P(O) = \frac{n(\langle Y, O\rangle \mid Y \in \{X_{-N}, \ldots, X_{-1}\})}{n(\langle Y\rangle \mid Y \in \{X_{-N}, \ldots, X_{-1}\})} \tag{4.2}$$

This recommender has his entire logic in stored procedures in the database.

4.1.5 Graph Partition Recommender

The *Graph Partition Recommender* computes an undirected graph based on the subsequences with the length two. The nodes can be either unweighted or weighted with the number of found subsequences in which the assigned navigation object occurred. The edges can also be unweighted or have the amount of found corresponding subsequences as their weight.

The graph is partitioned by the algorithm pmetis or kmetis from the METIS[KK] library in N partitions. Based on the navigation objects that occurred in the last T seconds the partition is selected that contains most of them. The navigation objects in the selected partition that not occurred in the last T seconds are recommended. They are descending ordered either by the supporting users count or by the number of the weights of the adjacent edges from all partitions or only the same partition. Figure 4.1 is showing how the same graph with and without node weights respectively edge weights is partitioned by the two algorithms algorithm.

The content of the graph file is created by stored procedures and the result is written directly into a file. After running the partitioning algorithm the computed partition labels are written back to the database. The recommendation logic is again placed in stored procedures in the database.

4.1.6 Spreading Activation Recommender

The *Spreading Activation Recommender* is based on the Spreading Activation Algorithm [And88]:

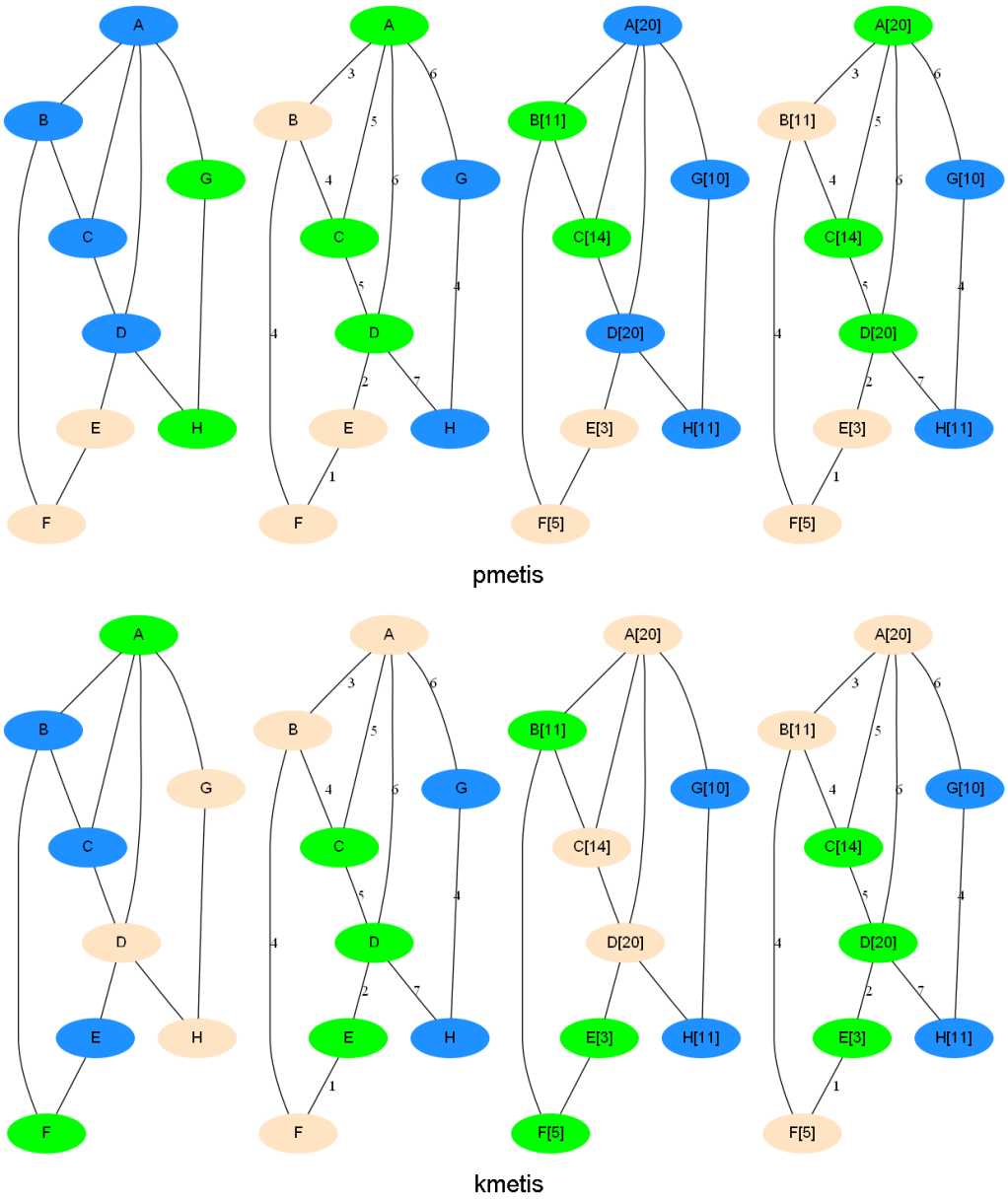


Figure 4.1: A graph partitioned by pmetis and kmetis

```
INPUT:  graph              G = (V, E)
        activation values  A[i]   in the range [0.0...INF[
        edge weights       W[i,j] in the range [0.0...1.0]
        firing threshold   F      in the range [0.0...1.0[
        decay factor       D      in the range ]0.0...1.0]

OUTPUT: activation values A[i] after spreading activation

while (exists unfired node V[i] having A[i] > F)
    foreach (unfired node V[i] having A[i] > F)
        mark V[ i ] as fired
        foreach (outgoing edge E[i,j] from V[i])
            set A[j] = A[j] + (A [i] * W [i,j] * D)
```

We use again the subsequences with the length two to create a directed graph. In contrast to the other algorithms this algorithm has his complete logic in the software an not in the database. The graph is stored in node and edge objects. Each node has a list the outgoing edges and a list of the incoming edges. The weights of the edges are the count of the found corresponding subsequences divided by the count of all found subsequences with the same first navigation object:

$$W[i,j] = \frac{n(\langle V[i].Obj, V[j].Obj\rangle)}{n(\langle V[i].Obj, O\rangle))} \tag{4.3}$$

There are several alternatives implemented which can be enabled by parameters:

- *directed* or *undirected graph* - when the undirected graph is selected then the undirected edges are created by union of the outgoing and incoming edges. The weights are calculated analogous to before:

$$W[i,j] = \frac{n(\langle V[i].Obj, V[j].Obj\rangle) + n(\langle V[j].Obj, V[i].Obj\rangle)}{n(\langle V[i].Obj, O\rangle)) + n(\langle O, V[i].Obj\rangle))} \tag{4.4}$$

- activation by *count* or by *time* - either the nodes assigned to the last N navigation objects or the navigation objects that occurred in the last T seconds are initially activated. The activation of a node is the assigning of the value 1.0, while the unactivated nodes remain at 0.0.

- spreading *for C iterations* or *until stabilised* - the algorithm can go on until there are no more activated nodes that can fire or only for C iterations.

- use *edge weights* or *decay factor* or *both* or *none* - the decaying of the activation value while spreading from one node to the next can be done by multiplying it with the edge weights, the decay factor or both. It is also to do spreading activation without decaying, but it has to be analysed whether this alternative produces useful results.

- *use firing threshold* or *not* - it is possible to do spreading activation without a firing threshold (or by setting it to 0.0). In this case a node can fire as soon as its activation value has been raised the first time.

- *simultaneous firing* or *"most activated" node first* - when a node fires it increases the activation value of its neighbors. Although a neighbor fires in the same iteration it would have a higher activation value than at the beginning of this iteration. To counteract this phenomenon we store the additions to the activation values in temporary fields and add them to the activation value after all nodes fired in one iteration. As an alternative we implemented the variation that in one iteration only the node with the highest activation value fires.

- *allow node retiring* or *not* - normally it is not possible that a node can be fired more than once. It could easily come to an endless firing sequence because each time when the activation value increases it sends more activation value to its neighbors. We implemented two alternatives where this is yet possible. Both are based on recursive propagation of the activation value. Not the actual activation value of the node is propagated to its neighbors, but only the decayed activation value of the firing predecessor.

  - *limit by iteration count* - we start the recursive activation value for all initially activated nodes with the iteration count as parameter. This parameter is decreasing for each call until it reaches zero. In this way we can limit the recursion depth and avoid endless firing sequences.
  - *fire once for each initially activated node* - in this alternative we forward the id of the initially activated node as a parameter and let each node fire only once for each initially activated node. In this manner each node can fire maximally K times, while K is the number of initially activated nodes.

- *return initially activated nodes* or *not* - after their firing the activation value of the initially activated nodes is set back to 0.0. But during the spreading it is probable that it can rise again due to firing of its neighbors. After the spreading algorithm terminates the recommender recommends the navigation objects in the descending order of the activation value of their associated nodes. The setting of this parameter determines whether the navigation objects that initially activated the nodes should be filtered or not.

How all this parameters affect the quality of the recommended resources must be evaluated. Figure 4.2 shows three iterations of the spreading activation algorithm on a directed graph with edge weights. We initially activated two nodes and used simultaneous firing with a decay factor 0.75 and a firing threshold 0.2.

4.2 Informational Goal

The algorithms supporting the informational goal can be separated in two parts: First the relevant terms must be extracted from the user context events and second resources must be selected based on the this terms. Since it is very hard to get a qualified analysis of the extracted terms which results in a quality number for the algorithms we decided to implement one term extraction algorithm and one resource recommendation algorithm.

4.2.1 Term Extraction

All events, except the keyboard events, which contain textual information represent this information with several token attributes per event. This attributes contain additionally to the term itself the count of the occurrences of this term in the textual data and the positions on which the terms were in the text. Our *Term Extraction Algorithm* creates a text object for each such event which contains the time when the event occurred and a list of the terms which occurred in this event with their counts. During this creation all words which can be found in an initially set stopword list are filtered.

Figure 4.3 describes the steps of the algorithm. The text objects are stored in a list which always contains the data of the last T seconds. When a term extraction is done then the terms data is accumulated in a list of terms with two numbers for each term: The sum of occurrences in all underlying text objects and the count of text objects in which this term

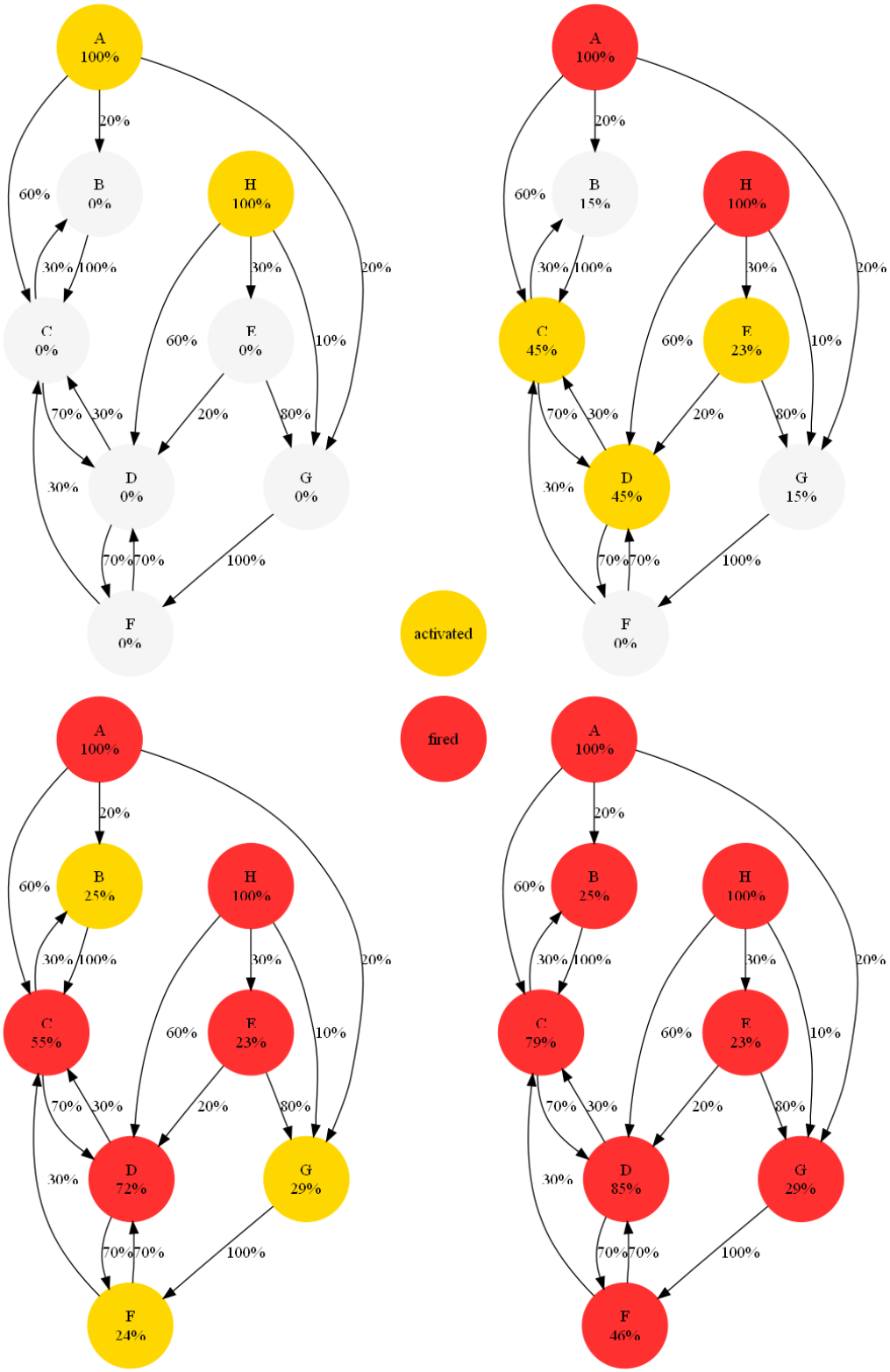


Figure 4.2: Three iterations of spreading activation

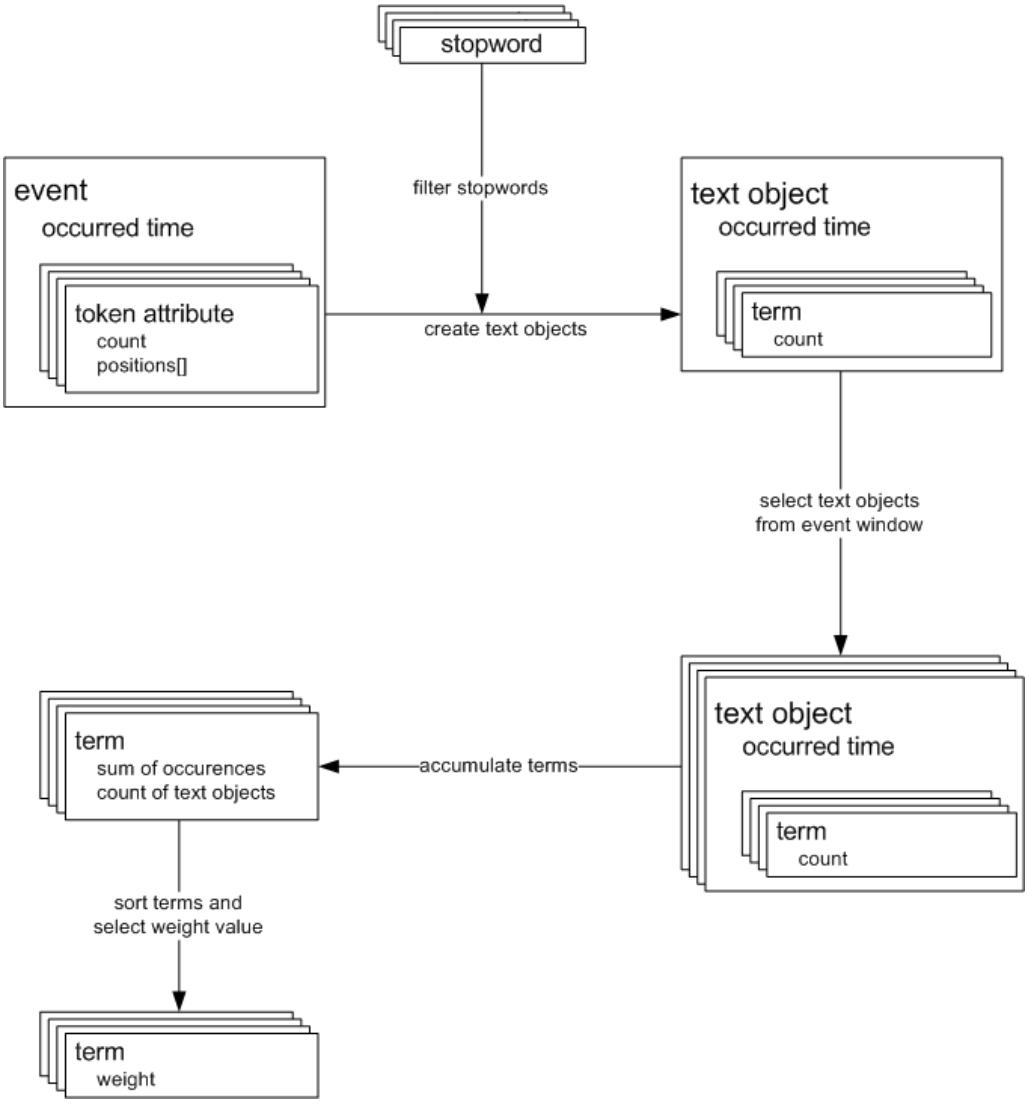


Figure 4.3: Steps of Term Extraction Algorithm

occurred. The list is sorted by one of this numbers and forwarded to resource recommendation part with only one weight value.

The algorithm can be configured by several parameters:

- *stopword selection type*, *count* N, *minimum percentage* P, and *language* - Our stopwords are selected from a table which contains the most occurred 1000000 words in the English and the German wikipedia articles. Each entry has beside a language field also a percentage value which is equal the count how often this word occurred in in all articles of the corresponding language divided by the sum of all words in all articles of the corresponding language. The entries are sorted by this percentage value. The stopword selection type sets whether the stopwords should be selected by their count (first N), by a minimum percentage (all with percentage value greater than P) or by both. The language sets whether the stopwords from both languages or from only one language should be used.
- *event window size* - it set how long the underlying event of a text object can date back so that this text object can be used for the term extraction. When the term extraction is started all text objects, whose time value is smaller that the time value of the actual event minus T seconds, are removed from the text object list.
- *extraction interval* - this parameter sets how often a term extraction should be make. To avoid problems with a timer and an additional thread we decided to use the incoming events as triggers and compared their time value to th time value of the event when the last extraction was made to decide whether it is time for the next extraction or not.
- *used event categories* - there are six boolean parameters which set whether the event from the categories `Foreground Window Changes`, `Application`, `Clipboard`, `Keyboard`, `Filesystem`, and `Filecontent` should be used or not.
- *file restriction regular expression* and *file filter regular expression* - this two regular expressions reduce the amount of filesystem and filecontent events that are used. Only the events are used whose path does match the file restriction regular expression and does not match the file filter regular expression.
- *term selection value* - this parameter sets whether the sum of occurrences or the count of text objects in which they occurred should be used as the weight by which extracted terms are sorted before forwarding to the resource recommendation.

4.2.2 Ressource Recommendation

Our *Ressource Recommendation Algoritm* is based on the pages tables of our English and German wikipedia databases. This tables have a `title` field from which we created an `extracted lower case terms` field which we used for a fulltext index. With this index we could fast search for pages which have a specific term as part of their title. The search run in three phases: The first phases takes three terms and searches all titles which contain each of the three terms in arbitrary order. Each found page is weighted with the sum of the three term weights. The second phase takes two terms and searches for titles which contain this two terms without a term between them. The weights of the found pages are set to the sum of the two term weights. The third phase takes on term and searches for titles which correspond exactly to the term. The weight of the term is used for the weights of the found pages. All found pages are sorted descending by their assigned weights and recommended to the user.

```
set found pages = {}
for (i = 2 to Min(term.Count, limitPhase1))
    for (j = 1 to i - 1)
        for (k = 0 to j - 1)
            select pages where
                term[i] in title AND
                term[j] in title AND
                term[k] in title
            set selected pages weights =
                term[i].weight +
                term[j].weight +
                term[k].weight
            add selected pages to found pages
for (i = 1 to Min(term.Count, limitPhase2))
    for (j = 0 to i - 1)
        select pages where
            (term[i] + term[j]) in title OR
            (term[j] + term[k]) in title
        set selected pages weights =
            term[i].weight +
            term[j].weight
        add selected pages to found pages
for (i = 0 to Min(term.Count, limitPhase3))
    select pages where
        term[i] = title
    set selected pages weights =
            term[i].weight
```

```
    add selected pages to found pages
sort found pages by weight descending
```

4.3 Transactional Goal

The algorithms supporting the transactional goal are based on the Machine Learning Library developed by Arne Beckhaus[Bec06]. The library contains several classifier algorithms which uses the stream of events to classify which task the user is currently working on. Additionally to a simple wrapper for one classifier algorithm from the library we implemented a voting classifier which makes a weighted vote over the classifications of several classifier algorithms to classify the current task. Both algorithms have in common that once they have classified the actual task they recommend the resources assigned to this task.

4.3.1 Classification Recommender

All classifiers in the machine learning library are based on a stream of instances that are created from the events. This stream is send through several preprocessors, which filter useless instances and split lists of terms combined in one instance into multiple individual instances. After being prepared the stream is sliced into windows of a designated length. Each window then represents a feature vector which contains the information whether the window contains a particular feature or not. This vector is submitted to the classifier algorithms which used their previously learned model to compute what is most probably the actual task the user is working on.

During the learning phase the algorithms are provided with labeled instances and are using N-Fold-Cross Validation[Rip96] to learn the best classifying model: The learning data is provided at once and separated into N parts. In N phases $N-1$ are used for learning a model and the remaining part is used for validating the quality of the learned model. After this N phases the average quality of all models is good representing value for the quality of this classifier.

The machine learning library contains the following classifier algorithms which all can be used for our classifier recommender:

- *Naive Bayes*

- *Descision Tree ID 3*
- *Descision Tree ID 3 with Post Pruning*
- *Euclidean Distance*
- *Irep*
- *Irep with just Splitting*
- *Irep without Prunning*
- *Sequential Minimal Optimization (SMO)*

A detailed description of this algorithms can be found in [Bec06].

To make the machine learning library usable for our application we first created lairing material in the expected format out of the event data stored in the database. Additionally we created a instance stream slicer that slices the instance stream into windows of an exact length. And finally we implemented an online instance stream which provides the classifier algorithm with instances directly from the events that are collected during the users work.

4.3.2 Voting Recommender

The Voting Recommender is making a majority vote over the classification of multiple Classification Recommender for the actual task. Since it is possible that we can get equal number of votes for multiple tasks we implemented a three decision level voting to ensure that we always can break the tie. The positions of the Classification Recommender in the provided list are also their weights. As long as we have a tie at a decision level we are going to the next level, but only with the votes that are in the tie at the current decision level. This are our three decision levels:

- *Maximum Count of Votes* - on this level the voting recommender classifies the actual task with the task that has received the maximum count of votes.
- *Minimum Sum of Weights* - on this level classification of the actual task is set to the task with the minimum sum of weights (out of the votes that have been in the tie the level before).

- *Lowest Single Weight* - on this level the tie from the level before is braked by classifying the task by the vote with the lowest weight. Since each weight is unique there can not be a tie after this decision level.

An example how this three decision level voting works is presented in Figure 4.4

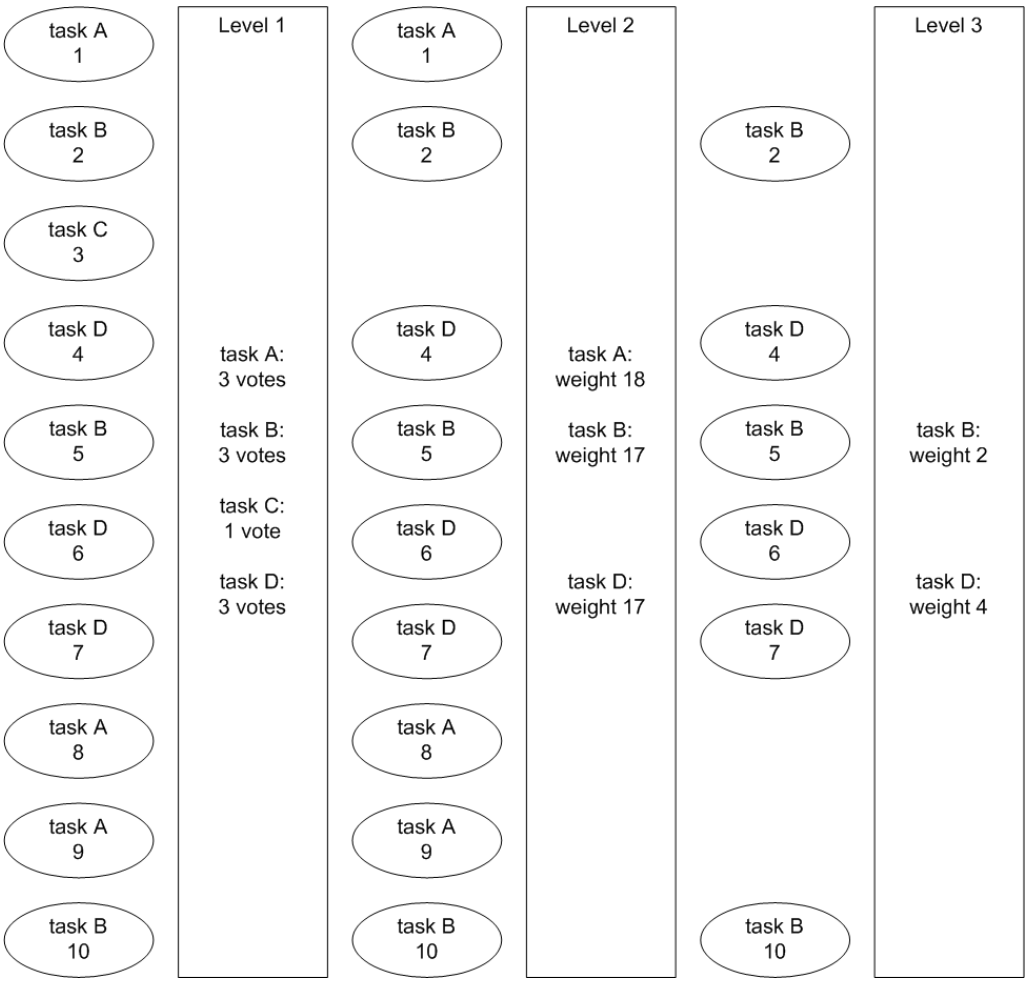


Figure 4.4: Example of Three Decision Level Voting

Chapter 5

Evaluation

In this chapter we describe the evaluation of our supporting system. First we defined *12 Tasks* which have be performed by *20 participants* during the first phase without any support to *collect training and evaluation data*. The *User Interaction Emulator* in combination with the collected data have been used to debug and to improve our algorithms by a *manual analysis*. As soon as the algorithms run stable we created several different configuration for each algorithm and used the *Automatic Analysis Tool* to find out which algorithm in which configuration is the best for each of the three user goals. In the second phase, the *Field Study*, we let 15 of the 20 participants perform the tasks a second time with support from our system to collect information about how our system helped them to fulfill the task faster and easier. After the second phase the participant filled out an *questionnaire* to collect their impressions of our supporting system.

5.1 Tasks

With the tasks that we selected for our evaluation we tried to cover a wide spectrum of the typical daily work of a knowledge worker. The tasks should be easy and short enough that every participant is able to complete them in about five minutes. The field study was designed to be finished in less than one hour for each participant.

- *Create a one slide presentation on Generics in Java*
  Create a very short PowerPoint-presentation pres.ppt about how to use Generics in Java. You might want to investigate on Generics in the Internet first.

- *Update the SRN page of your Project*
  Since your job is also to initiate collaboration between projects you always have to update the SAP Research Net webpage with your current focus of interest. Check for the correctness of information included in your projects SRN page and update the website if necessary.

- *Distribute Presentation Slides*
  A college requests the slide series of presentation you recently held. Find the slides in the folder Slides and send the content of the folder to the mailing list DL SAP Research APOSDLE Evaluation.

- *Visualization of Research Results*
  You just received the research results of your student assistant. The document research.xls containins the global revenue for showering gel. You want to visualize the year sums graphically in a diagram.

- *Leave request*
  Find out on which date is easter next year. You might want to create a leave request in the SAP Portal for the week before easter. Check out in the Leave Request Overview whether you still have enough days for vacation to book the whole week.

- *Translation of Executive Summary*
  Your project partner supplied a short description of his activities in the last year. You have to translate the document in order to forward it to the European Commission. Open the file original.txt. It contains a German text. Translate it to English and save the translation in the file translation.txt.

- *Prototype Development: Hello World in Java*
  Knowledge work sometimes includes software development activities. In your work plan there is a task to create a software prototype for a demonstration. Write a java-program hello.java that prints out the words "Hello world" to fulfill this task.

- *Inventory Update*
  Suddenly your work process is interrupted by a request of your location manager. You should help him to update the local inventory. Therefore you have to write an email with the Equipment Number of your computer to Matthus Martynus.

- *Handout*
  You have to hold your presentation testing.ppt tomorrow. Create a short handout testing.doc with a few key facts of this presentation.

- *UML*
  Create an UML-diagram of the following class: The name is SAP-Worker, a subclass of person. It has the attributes name and position. It should be stored as a picture in UML.*. (You can choose the type of the picture.)

- *Budget Calculation*
  For your project a budget of 10.000 euro has been assigned. You and your two co-workers will get 1.234 euro each of it. You will need seven students for the evaluation and each of them will cost you 789 euro. How much money will be left for the catering at the final presentation? Save the calculation in calculation.*. (You can choose the type of this file.)

- *Software Update*
  The IT Security department kindly asked you to update the software on your computer. Please check in the SAP Software Corner, whether there is a newer version of SnagIT than the one installed on your computer. In case, please update the outdated version.

5.2 Data Collection

During the data collection phase we let each of our participant perform the 12 tasks. We stored the collected event data in the database and labeled it with the task during which it was collected. To avoid correlations between the tasks the participant got the tasks in a random order. To make the data more suitable for the algorithms we specified the file names and one folder as location for all files created or needed during the tasks.

5.3 Manual Analysis

During the manual analysis phase we tested, adjusted, and evaluated the implemented algorithms. The User Interaction Emulator replaced the Context Monitor and fed the User Interface with event data. The content of the send events was shown in the emulator so that it was possible to see how the algorithms react on which data. Since each event is labeled with the time when it was collected and the algorithms use this data instead of the time when they receive the event. In this manner it was possible to send the events uncontinously and to see the impact of each each individual event on the analysed algorithms.

After debugging the algorithms and finding some influencing parameters we ran the emulator in real-time mode and analysed what the evaluation participant would have seen if the recommender algorithms would have run during the data collection phase. The shown event data gave us hints about the participants intention and we could analyse how the recommended resources would fit to it.

5.4 Automatic Analysis

During the automatic analysis we ran our algorithms with more than 3000 different configurations for their influencing parameters. To show the influance of an parameter we took all results for an algorithm and grouped them all used values for this parameter. We used special candlestick charts to visualize the results: The vertical lines show the distribution by starting at the minimal reached value and ending at the maximal reached value, while the short horizonatl line show the average reached value.

For the analysis of the support for the navigational goal and the transactional goal we also compared the results the different algorithms to each other. For the parameters they have in common we compared the influance of this parameters to the numbers reached for complete analysis data. Additionally we compared the best configuretions for each algorithm to each other on task basis: It can be seen how good each of the algorithms worked during each of the twelve tasks.

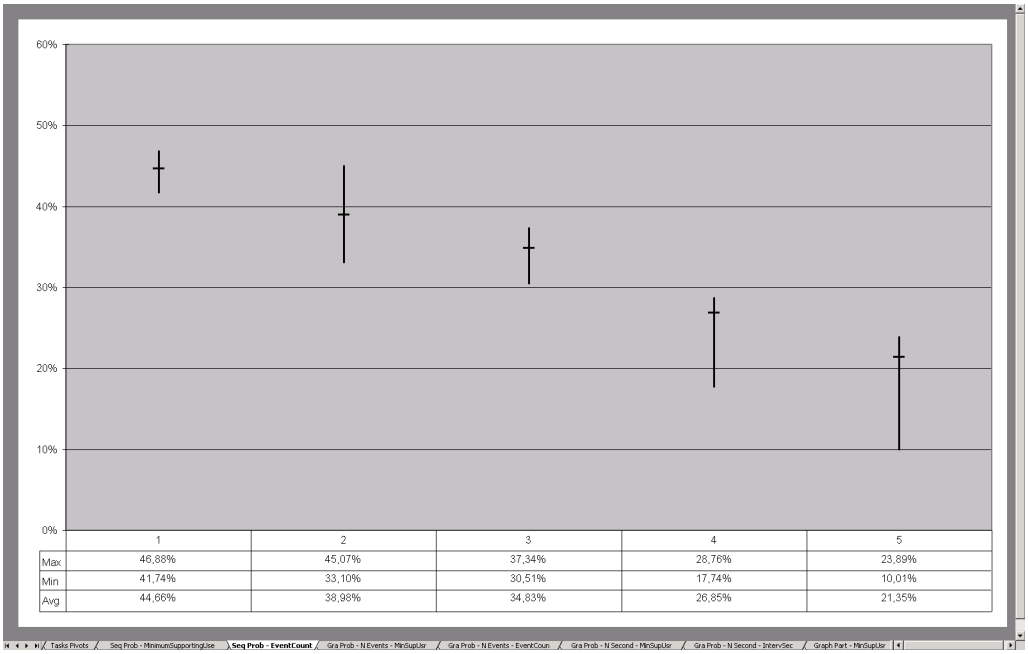
5.4.1 Analysis of Recommended Navigational Objects

For the algorithms supporting the navigational goal we analysed how good the recommended navigational objects match to the navigational objects exracted from the events. For each extracted navigational object we looked up whether this object could be found in the top 20 positions of the recommendet list before his event has been propagated. We performed a N-fold-cross analysis: The session beeing analysed was excluded from the database on which the recommendations of the the algorithms were based on.

Sequence Probability Recommender

For the Sequence Probability Recommender we analysed the influance of the following parameters:

- *Navigation Object Count* - Values: 1, 2, 3, 4, 5
- *Minimum Supporting Users* - Values: 1, 2, 3, 4, 5, 7, 10, 15



	1	2	3	4	5
Max	46,88%	45,07%	37,34%	28,76%	23,89%
Min	41,74%	33,10%	30,51%	17,74%	10,01%
Avg	44,66%	38,98%	34,83%	26,85%	21,35%

Figure 5.1: Sequence Probability Recommender - Influence of Event Count

Figure 5.1 is showing the influance of the *Navigation Object*: The more navigation objects are used as input the lower is the found percentage. Additionally it can be observed that the average of the found percentages is much nearer to the maximum than to the minimum what is an indication for a few significantly lower found percentages for the some configurations with higher Navigation Object Count. A higher Navigation Object Count results in a longer sequence beeing searched in the database and it is more probable that the sequence is not present in the database or none of the few occurrences of it is followed by the actual navigation object.

The numbers visualised in Figure 5.1 show that the amount of *Minimum Supporting Users* has only a small influance on the found percentages: From one supporting user to ten the average found percentages are slowly going up, but also the spreading. The maximum is going up but the minimum is going down so we can argue that the configurations with less minimum supporting users are more reliable but with lower aaccuracy. The numbers for minimum 15 supporting users show that it is not efficient to use as much minimum supporting users as possible.

In combination with Figure 5.1 it can be stated that the higher Navigation Object Count and high Minimum Supporting Users are the worst

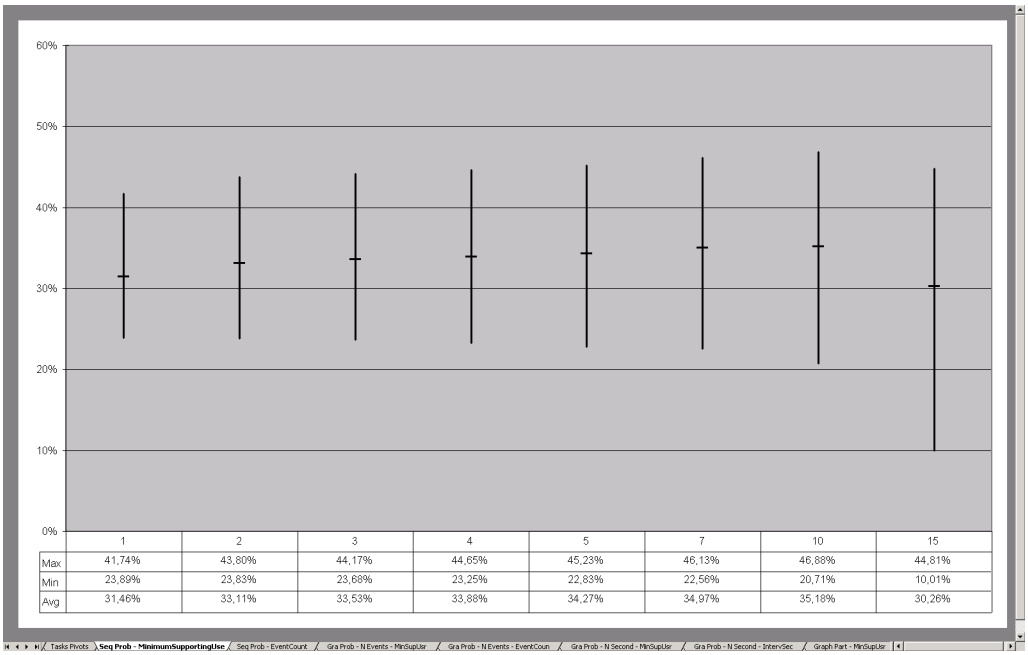


Figure 5.2: Sequence Probability Recommender - Influence of Minimum Supporting Users

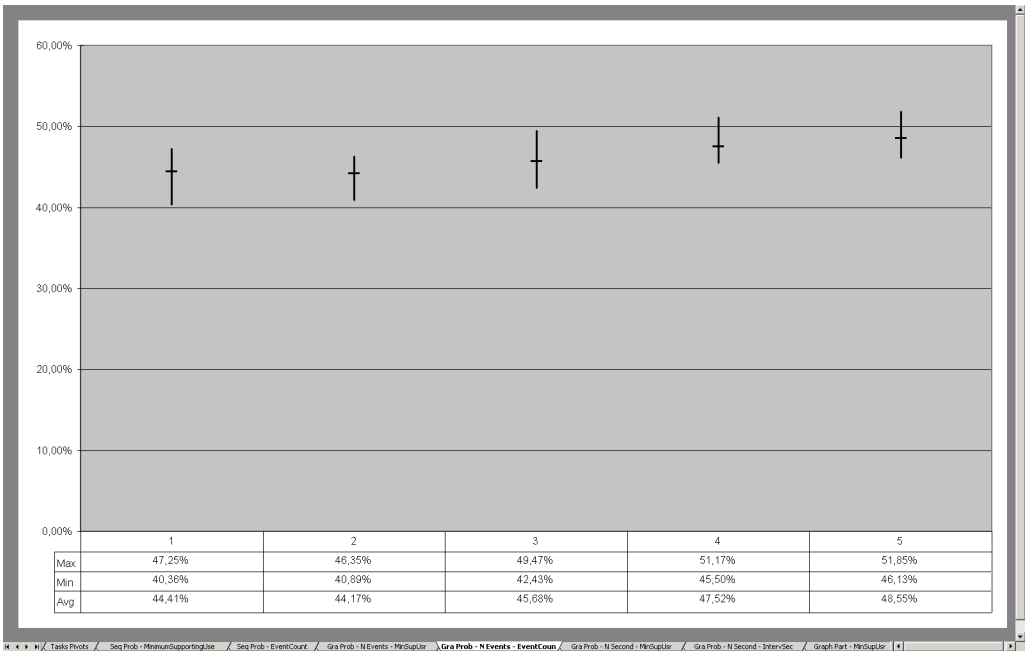
configuration for the Sequence Probability Recommender, while the combination of a "Single Navigation Object Sequence" with minimum ten supporting users results in the highest found percentage.

Graph Probability Recommender

For the Graph Probability Recommender we have two different kind of navigation object selection with their own sets of parameters parameters:

- *Navigation Object Count Based Selection*
  - *Navigation Object Count* - Values: 1, 2, 3, 4, 5
  - *Minimum Supporting Users* - Values: 1, 2, 3, 4, 5, 7, 10, 15
- *Time Interval Based Selection*
  - *Interval Seconds Count* - Values: 30, 60, 90, 120
  - *Minimum Supporting Users* - Values: 1, 2, 5, 10, 15

Figure 5.4 is showing the influance of the *Navigation Object Count* on the Graph Probability Recommender. It can be observed that with higher Event Count the found percentages are also going up. That is reasonable



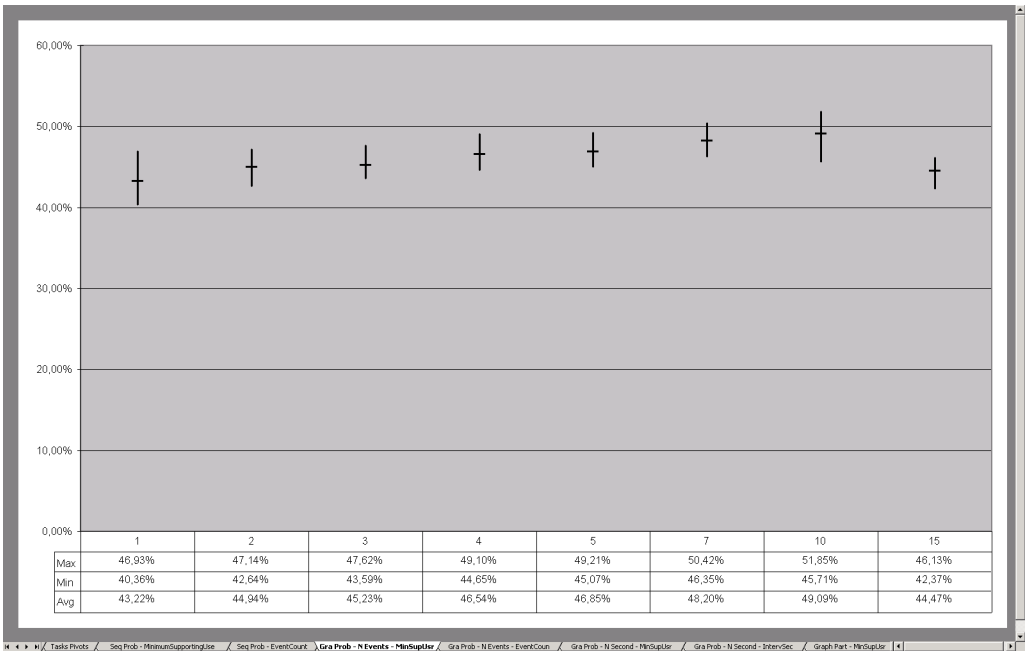
	1	2	3	4	5
Max	47,25%	46,35%	49,47%	51,17%	51,85%
Min	40,36%	40,89%	42,43%	45,50%	46,13%
Avg	44,41%	44,17%	45,68%	47,52%	48,55%

Figure 5.3: Graph Probability Recommender Last N Events - Influence of Event Count

since more used navigation objects result in more nodes in the graph whose edge probabilities will be accumulated and in this way multiple nodes can support one common neighbour which stands for a navigation object which often follows multiple of the previously extracted navigation objects. The found percentages for one navigation object are the the same as for one navigation object at the Sequence Probability Recommender since searching for a "Single Navigation Object Sequence" is the same as using the probabilities of the edges of a single node in the directed navigation object graph.

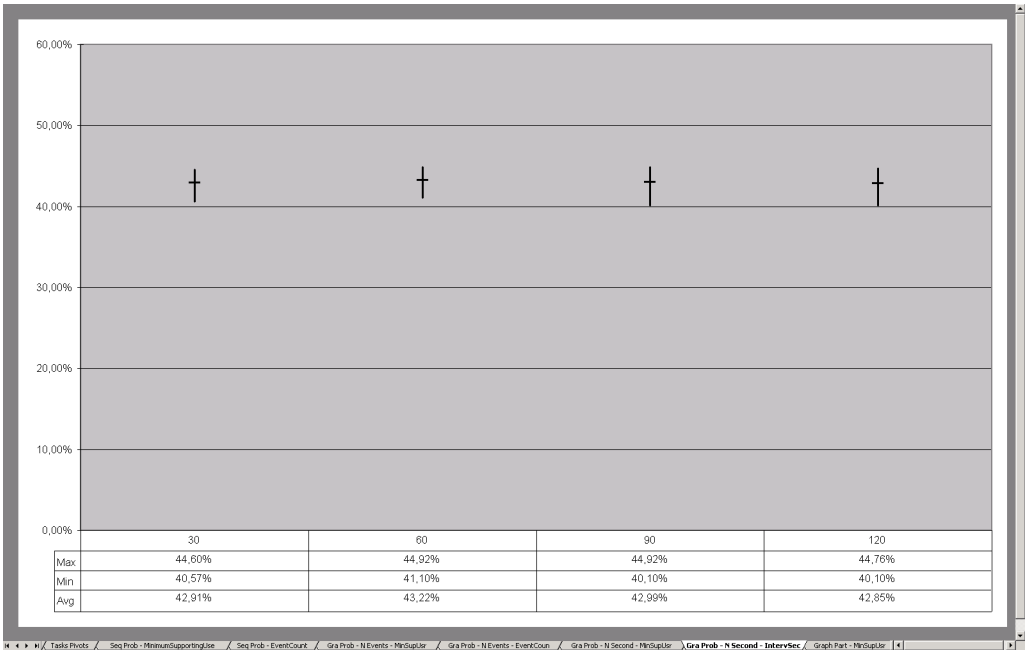
In Figure 5.4 the effect of more *Minimum Supporting Users* on the found percentage of the Graph Probability Recommender with *Navigation Object Count Based Selection* of navigation objects. The found percentages are going up with more minimum supporting users, but only up to ten; for 15 the numbers are going significantly down again.

Figures 5.5 and 5.6 are showing the influance of the parameters *Interval Seconds Count* and *Minimum Supporting Users* on the found percentages of the Graph Probability Recommender with *Time Interval Based Selection* of navigation objects. It is obvious that the length of the interval has no influance on the found percentages and they only depend on the amount of Minimum Supporting Users: Again the found percentages are rising with more minimum supporting users up to ten and falling for 15.



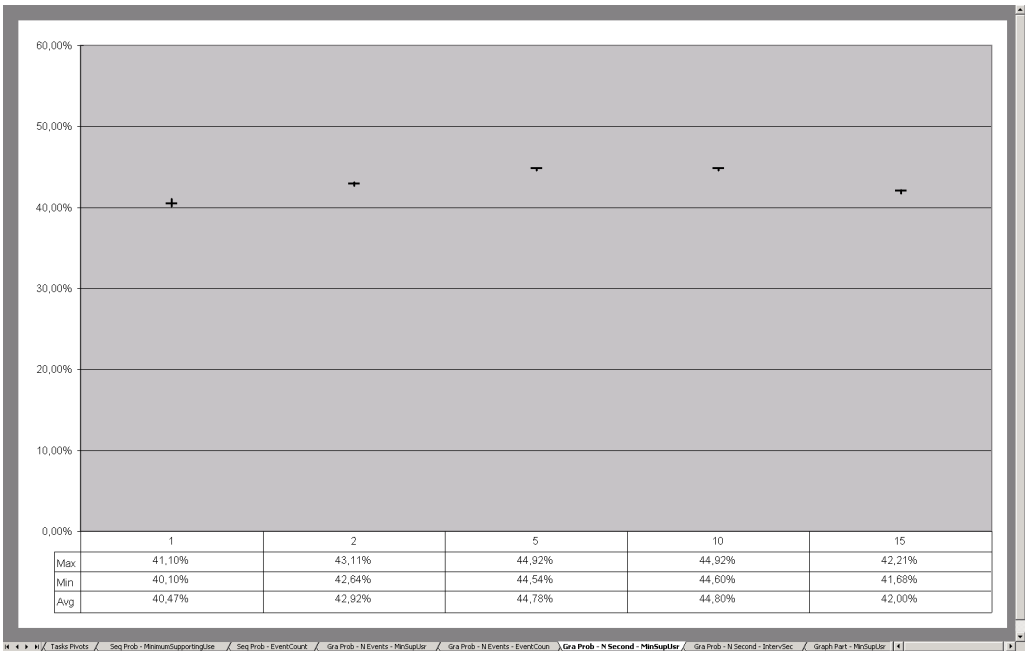
	1	2	3	4	5	7	10	15
Max	46,93%	47,14%	47,62%	49,10%	49,21%	50,42%	51,85%	46,13%
Min	40,36%	42,64%	43,59%	44,65%	45,07%	46,35%	45,71%	42,37%
Avg	43,22%	44,94%	45,23%	46,54%	46,85%	48,20%	49,09%	44,47%

Figure 5.4: Graph Probability Recommender Last N Events - Influence of Minimum Supporting Users



	30	60	90	120
Max	44,60%	44,92%	44,92%	44,76%
Min	40,57%	41,10%	40,10%	40,10%
Avg	42,91%	43,22%	42,99%	42,85%

Figure 5.5: Graph Probability Recommender Last T Seconds - Influence of Interval Seconds



	1	2	5	10	15
Max	41,10%	43,11%	44,92%	44,92%	42,21%
Min	40,10%	42,64%	44,54%	44,60%	41,68%
Avg	40,47%	42,92%	44,78%	44,80%	42,00%

Figure 5.6: Graph Probability Recommender Last T Seconds - Influence of MinimumSupportingUsers

Overall it can be stated the Graph Probability Recommender has his highest found percentage with a *Navigation Object Count Based Selection* of navigation objects with five navigation Objects and a minimum of ten supporting users.

Graph Partition Recommender

For the Graph Partition Recommender we analysed the influance of the following parameters:

- *Interval Seconds Count* - Values: 30, 60, 90, 120
- *Minimum Supporting Users* - Values: 1, 2, 5, 10, 15
- *Partition Count* - Values: 5, 10, 15, 20

The following parameters remained unchanged, since they showed no influance during the manual evaluation phase:

- *Graph type* - weighted nodes and edges
- *partition algorithm* - pmetis

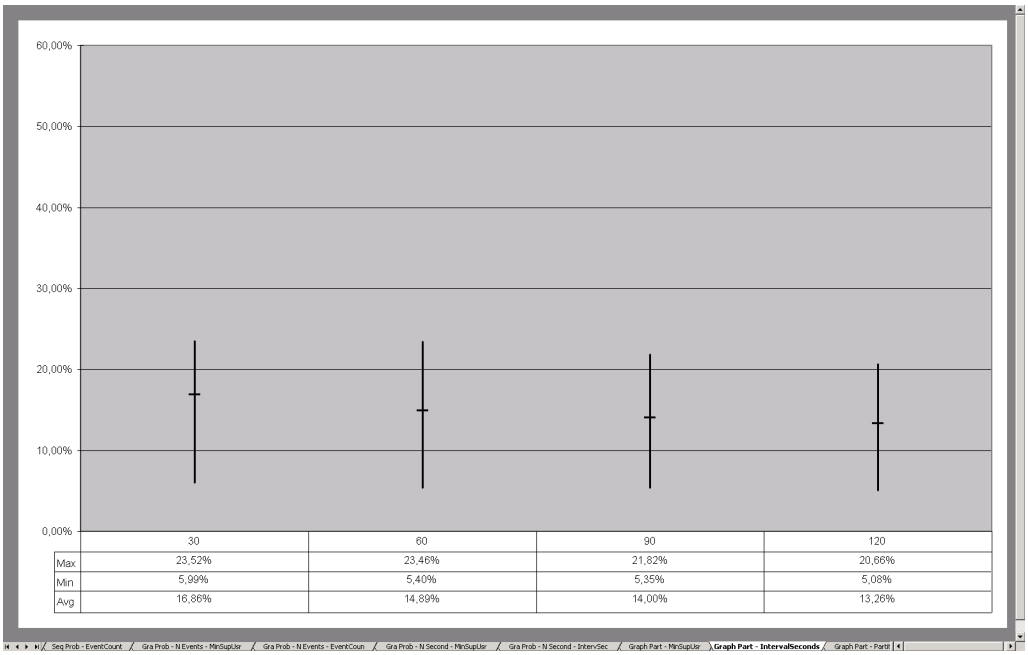


Figure 5.7: Graph Partition Recommender - Influence of Interval Seconds

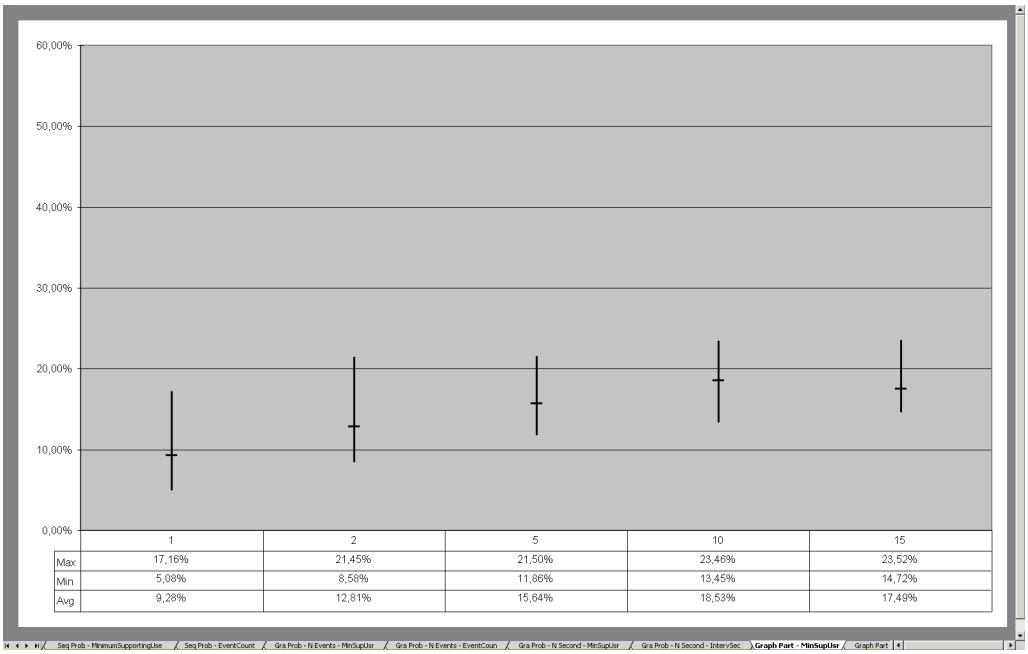
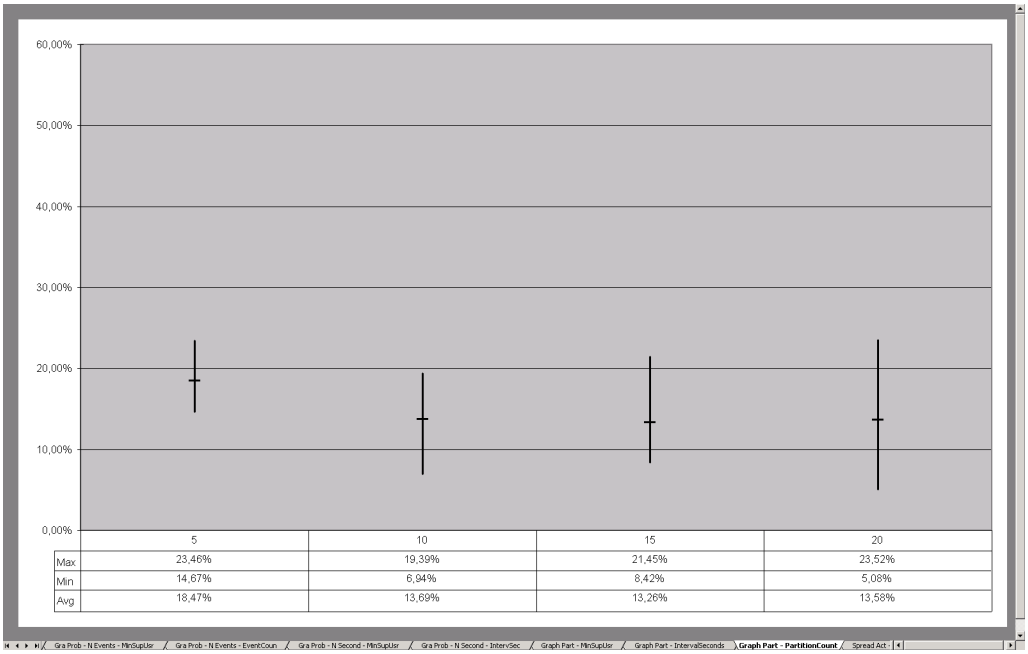


Figure 5.8: Graph Partition Recommender - Influence of Minimum Supporting Users

- *Navigation Object Ordering* - by all adjacent edges

Figure 5.7 shows that the the *Interval Seconds Count* has only a very small influance. With a longer interval all numbers are getting a little bit worse. In contrast to this Figure 5.8 shows significant influance of the parameter *Minimum Supporting Users*: With a higher amount the average found percentages are getting better and there is less spreading. The average found percentages again rise until ten users and become a little bit worse for 15 users.



	5	10	15	20
Max	23,46%	19,39%	21,45%	23,52%
Min	14,67%	6,94%	8,42%	5,08%
Avg	18,47%	13,69%	13,26%	13,58%

Figure 5.9: Graph Partition Recommender - Influence of Partition Count

With the parameter *Partition Count* Figure 5.9 shows the influance of the only one investigated algorithm specific parameter. For five partitions the algorithm reaches the best found percentages in average, minimum, and maximum. For more partitions the average found percentages stay stable at a much lower value, while there is more spreading.

Overall the best configuration for the Graph Partition Recommender was an 30 second interval with a minimum of 15 supporting users and five partitions. The reason for this is that with a high demand of minimum supporting users the graph becomes more sparse and only the important navigation objects remain that were used by almost every user. Additionally the low number of partitions combines navigation objects used for multiple similar tasks into one partition so that it is more probable to select the right partition: One significant navigation object (for example the start

of an application) can be used in all tasks whose navigation objects are combined in the selected partition.

Spreading Activation Recommender

For the Graph Partition Recommender we analysed the influance of the following parameters:

- *Activated Navigation Object Selection Type*
  - *Navigation Object Based Selection* - Values: 1, 3, 5, 10
  - *Time Interval Based Selection* - Values: 30, 60, 120
- *Decaying Type*
  - *Decay Factor 0.5*
  - *Decay Factor 0.75*
  - *Edge Probabilities*
- *Firing Threshold* - Values: 0.01, 0.1
- *Graph Type* - undirected or directed
- *Minimum Supporting Users* - Values: 2, 7, 15
- *Spreading Type*
  - *For 5 Iterations*
  - *For 10 Iterations*
  - *For 15 Iterations*
  - *Until Stabelised*
- *Recommending of Initially Activated Navigation Objects* - filter or recommend

We investigated only *simultaneous firing without node refiring*, since the alternatives with *firing "most activated" node first* and *multiple firing per node* resulted in much higher calculation time with not better recommendations.

In Figure 5.10 we display the found percentages depending on the parametrised *Activated Navigation Object Selection Type*: It can be observed that for the *Navigation Object Count Based Selection* the best results were produced

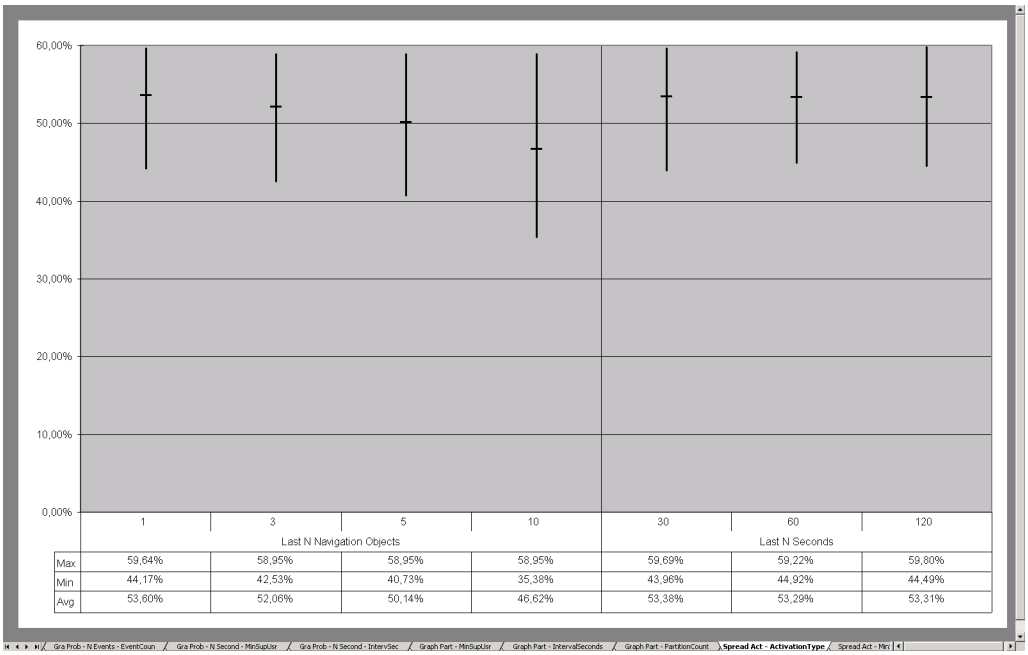


Figure 5.10: Spreading Activation Recommender - Influence of Activation Type

with the single last extracted navigation object. The more navigation objects have been used the lower the found percentages have become. In contrast to this the seconds count for the *Time Interval Based Selection* has nearly no influance on the found percentages. Most intresting is that the activation of the single last extracted navigation object results in the same found percentages as the activation of all navigation objects extracted in the last 30, 60, or even 120 seconds.

Figure 5.11 shows the influance of the *Decaying Type*, the *Firining Threshold* and the *Graph Type* It can be seen that none of them has any infuance on the found percentages. The *Minimum Supporting Users* have an influance on the found percentages as well for *Navigation Object Count Based Selection* as for the *Time Interval Based Selection.* for both the average found percentages get better with mor minimum supporting users, but there is less spreading for a minimum of seven supporting users than for a minimum of 15 supporting users.

In Figure 5.13 we visualised teh influance of the *Spreading Type* and the *Recommending of Initially Activated Navigation Objects.* As it can be clearly seen they both have had no influance on the found percentages.

Overall the best configuration for Spreading Activation Recommender was a *Time Interval Based Selection* for an interval of 120 seconds, with edge probability depending decaying type, a firing threshold of 0.01, on a

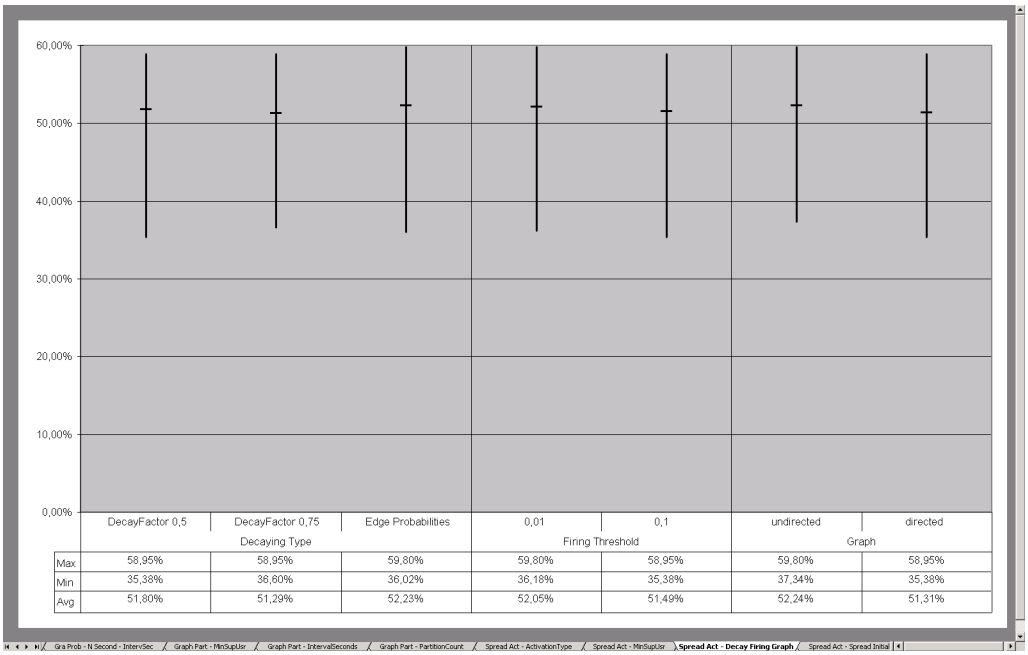


Figure 5.11: Spreading Activation Recommender - Influence of Decaying Type, Firing Threshold, and Graph Type

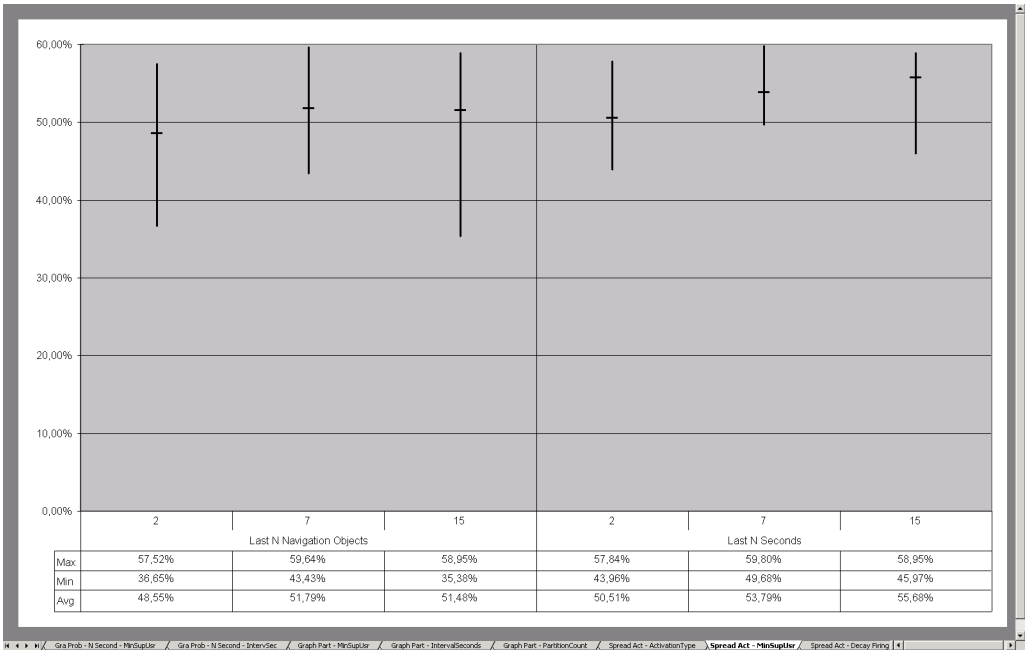


Figure 5.12: Spreading Activation Recommender - Influence of Minimum Supporting Users

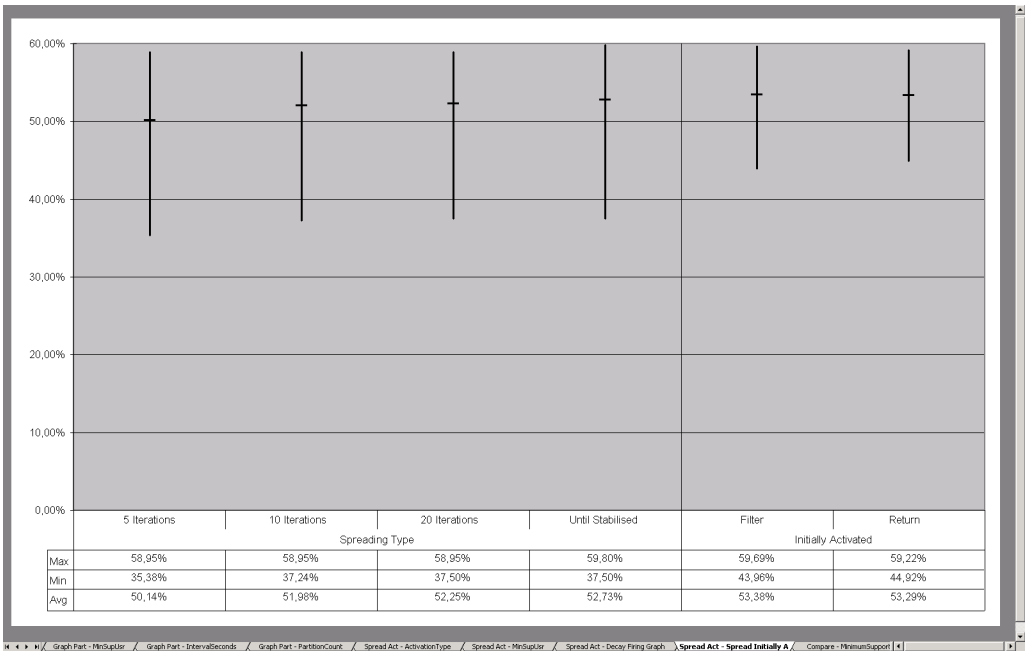


Figure 5.13: Spreading Activation Recommender - Influence of Spreading Type and Recommending of Initially Activated Navigation Objects

directed graph with a spreading until stabalised and a minimum of seven supporting users.

More experimental results can be found in the appendix but are not described in more detail here.

5.5 Field Study

During the *Field Study* 15 of the initiall 20 participants fullfilled the twelve tasks a second time. To avoid the influance of remebering already done tasks betwwen the two evaluation phases alapsed about four weeks.

In the Figures 5.14 to 5.16 we visualize the logs for the evaluation sessions of the participants. The green and the red areas show whether the transactional recommender classified the task correctly at this time. the three dotted lines represent the three used recommedners: The lowest stands for the *Transactional Recommender*, the middle for the *Navigational Recommender*, and the highest for the *Informational Recommedner*. Each black point on it represents a click on one of the recommended resources. The other colors mark the lots of each of the tasks on the complete sessions. The sessions are normalized, so that the width of the task lots displays their precentage on the session duration.

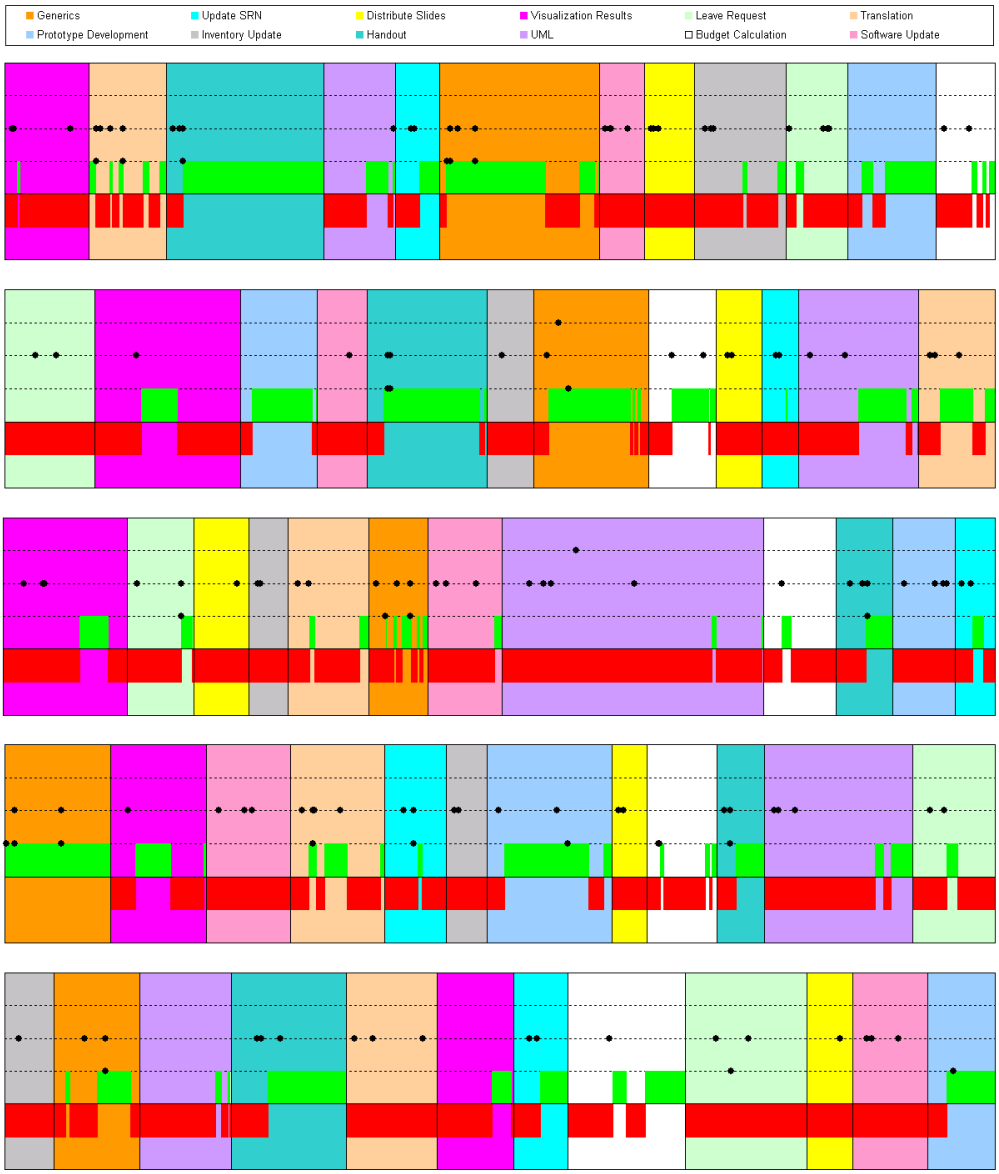


Figure 5.14: Field Study Results - Session Logs(1)

Figure 5.15: Field Study Results - Session Logs(2)

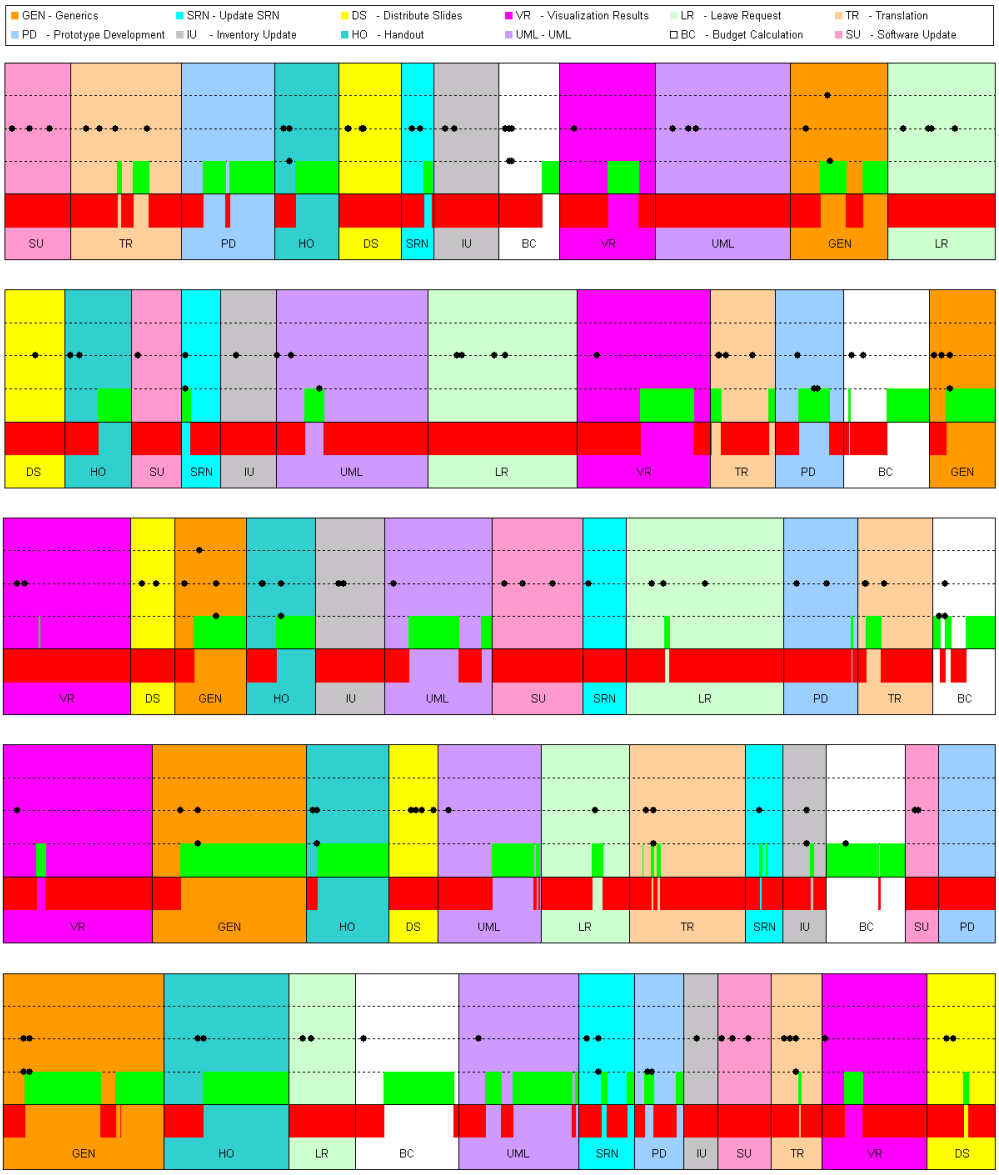


Figure 5.16: Field Study Results - Session Logs(3)

It can be claerly obsereved that most of the used resources were recommended by the navigational recommender. And only a few by the informational recommender. The classification accuracy were about 50 perecent and the lots of the tasks are very different for each of the participants.

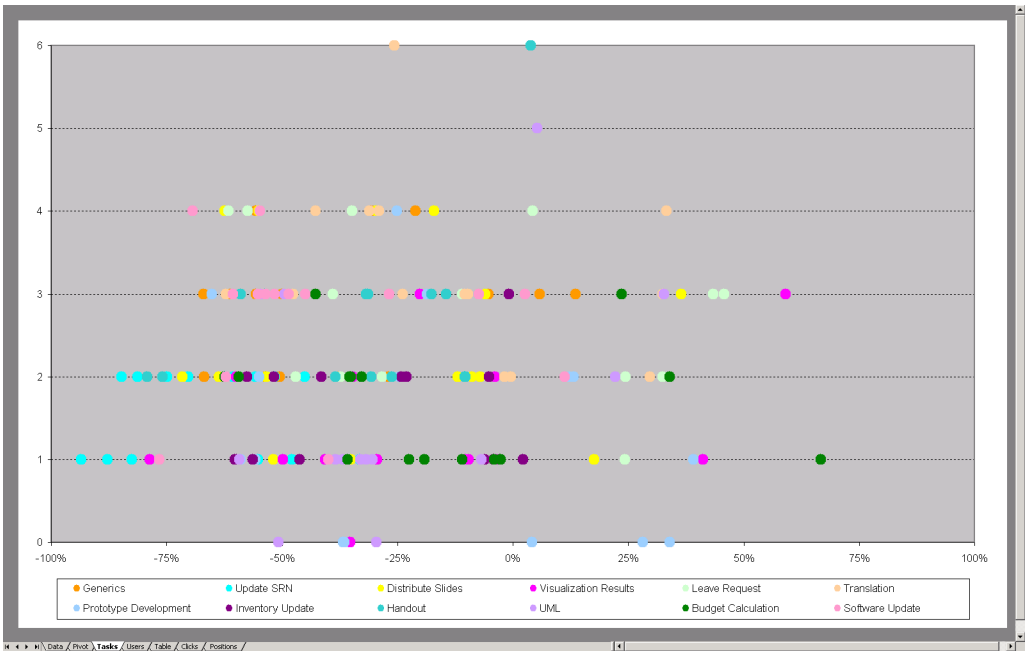


Figure 5.17: Field Study Results - Relation Clicks to Duration Change - By Tasks

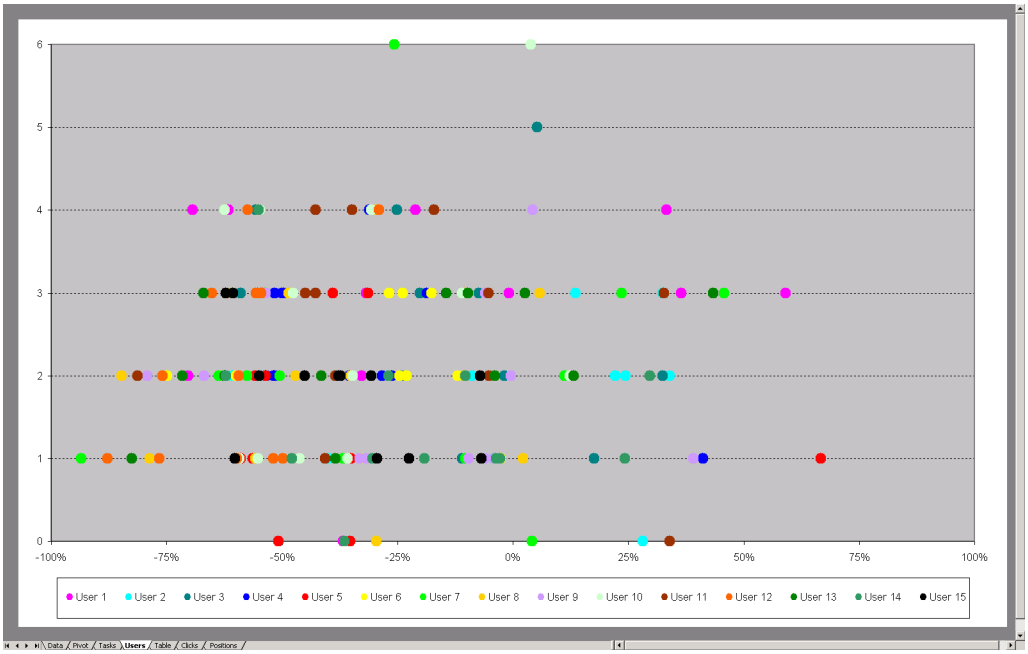


Figure 5.18: Field Study Results - Relation Clicks to Duration Change - By Participants

	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	Complete
User 1	-21,05% (4)	-70,31% (2)	36,42% (3)	59,05% (3)	-61,63% (4)	33,33% (4)	-36,76% (0)	-0,85% (3)	-31,76% (3)	-39,92% (1)	-32,75% (2)	-69,38% (4)	-29,96% (33)
User 2	13,59% (3)	-60,63% (2)	-8,89% (2)	158,20% (1)	24,38% (2)	32,52% (3)	28,13% (0)	-4,19% (1)	105,93% (2)	22,20% (2)	33,93% (2)	-39,89% (1)	14,97% (21)
User 3	-55,59% (4)	-62,33% (2)	17,65% (1)	-20,14% (3)	32,45% (2)	-1,74% (2)	-25,12% (4)	-24,14% (2)	-58,91% (3)	5,31% (5)	-10,97% (1)	-7,33% (3)	-18,91% (32)
User 4	-50,00% (3)	-51,56% (2)	-53,40% (2)	41,22% (1)	-28,30% (2)	-31,00% (4)	-18,57% (3)	-51,75% (2)	-26,24% (2)	-49,53% (3)	-35,41% (2)	-51,56% (3)	-33,69% (29)
User 5	-53,55% (2)	-55,81% (2)	-35,03% (1)	-35,27% (0)	-39,01% (3)	-54,22% (3)	-58,88% (1)	-56,33% (1)	-31,30% (3)	-50,72% (0)	66,67% (1)	-55,17% (3)	-42,78% (20)
User 6	-61,19% (3)	-74,88% (2)	-11,97% (2)	-59,88% (2)	-29,05% (4)	-23,87% (3)	-24,51% (2)	-22,97% (2)	-17,65% (3)	-33,05% (1)	-2,67% (1)	-26,83% (3)	-32,13% (28)
User 7	-50,48% (2)	-93,52% (1)	-63,70% (2)	-10,33% (1)	45,71% (3)	-25,65% (6)	4,22% (0)	-57,58% (2)	-53,72% (3)	-36,73% (1)	23,60% (3)	11,22% (2)	-31,92% (26)
User 8	5,83% (3)	-84,77% (2)	-29,53% (4)	-78,69% (1)	-47,00% (2)	-10,39% (3)	-55,81% (1)	2,16% (1)	-10,42% (2)	-29,54% (0)	-35,14% (2)	-48,55% (3)	-41,62% (24)
User 9	-66,88% (2)	-33,11% (1)	-6,02% (3)	-9,50% (1)	4,33% (4)	-0,48% (2)	39,08% (1)	-6,15% (1)	-79,15% (2)	-31,79% (1)	-4,03% (1)	-53,47% (3)	-25,14% (22)
User 10	-30,60% (4)	-55,26% (1)	-62,41% (4)	-34,72% (2)	-10,90% (3)	-47,62% (3)	12,39% (2)	-46,19% (1)	3,83% (6)	-59,27% (1)	-35,78% (1)	-54,82% (4)	-32,17% (32)
User 11	-5,29% (3)	-81,25% (2)	-16,99% (4)	-40,67% (1)	-34,84% (4)	-42,79% (4)	33,97% (0)	-5,11% (2)	-38,39% (2)	32,76% (3)	-42,66% (3)	-45,00% (3)	-29,77% (31)
User 12	-55,65% (3)	-87,85% (1)	-51,92% (1)	-49,76% (1)	-57,36% (4)	-29,05% (4)	-65,16% (3)	-59,71% (1)	-75,80% (2)	-54,53% (3)	-59,42% (2)	-76,53% (1)	-58,82% (26)
User 13	-67,00% (3)	-82,46% (1)	-71,63% (2)	-3,88% (2)	43,42% (3)	-9,74% (3)	13,22% (2)	-41,52% (2)	-14,38% (3)	-38,44% (1)	-62,46% (2)	2,58% (3)	-31,98% (27)
User 14	-26,97% (2)	-47,92% (1)	-55,06% (4)	-3,61% (1)	24,22% (1)	29,72% (2)	-36,50% (0)	-2,82% (1)	-10,26% (2)	-30,43% (1)	-19,18% (1)	-62,17% (2)	-20,00% (18)
User 15	-37,77% (2)	-45,13% (2)	-7,14% (2)	-29,46% (1)	-37,33% (2)	-62,14% (3)	-54,94% (2)	-60,16% (1)	-30,70% (2)	-6,83% (1)	-22,44% (1)	-60,66% (3)	-36,32% (22)
Average	-37,51% (3)	-65,79% (2)	-27,97% (2)	-7,83% (1)	-11,39% (3)	-16,21% (3)	-16,35% (1)	-29,15% (2)	-24,60% (3)	-26,70% (2)	-15,91% (2)	-42,50% (3)	-30,02% (26)

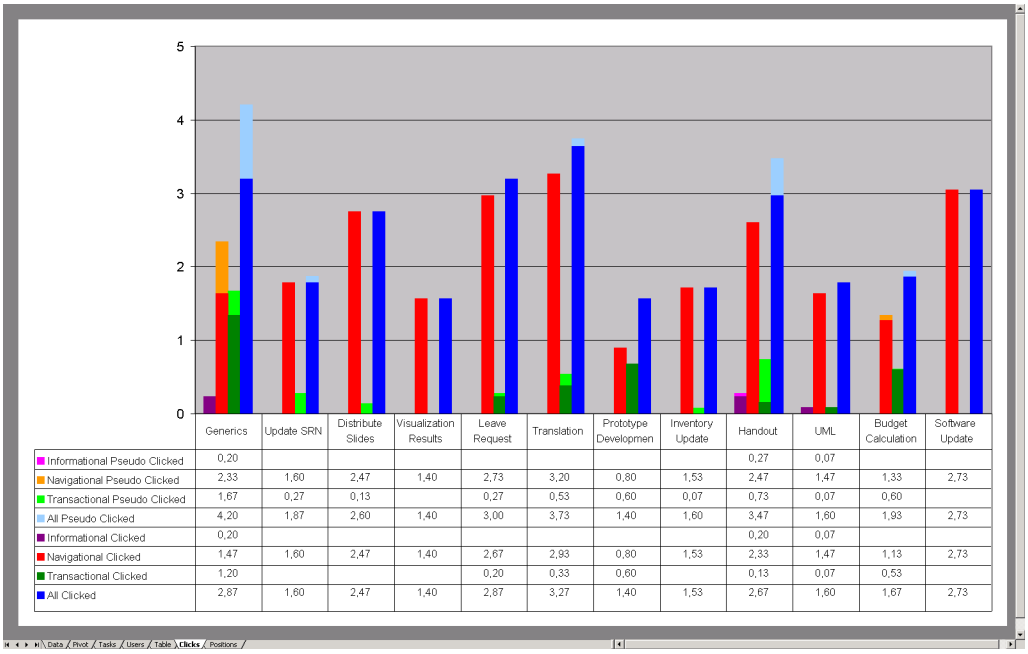
Figure 5.19: Field Study Results - Relation Clicks to Duration Change - Table

Figures 5.17 to 5.19 summarize the corelation between used resources and reduction of needed time for eahc of the tasks. In 5.17 the dots are cloured by tasks and in 5.18 by participants: In none of them two a grouping of a color can be observed. By none of the user and during no task a significant high number of clicks or high reduction of time is showed. Figure 5.19 shows the numbers for the other two diagramms: It is interesting to see that all users saved in average nearly two third of their time during the "Update SRN" task and that only user 2 needed more time for the tasks during the second phase, but he was very interested in the theory of the system.

Finally Figures 5.20 and 5.21 show how many resources have been used in average during the tasks and on which position of the lsit tha have been clicked. The numbers confirm the observations from the session logs that most of the used resources come from the navigational recommender. But it can also be clearly seen that the average position is much lower for the transactiona recommender, what is obvious since this recommender used short predefined lists of usefull resources.

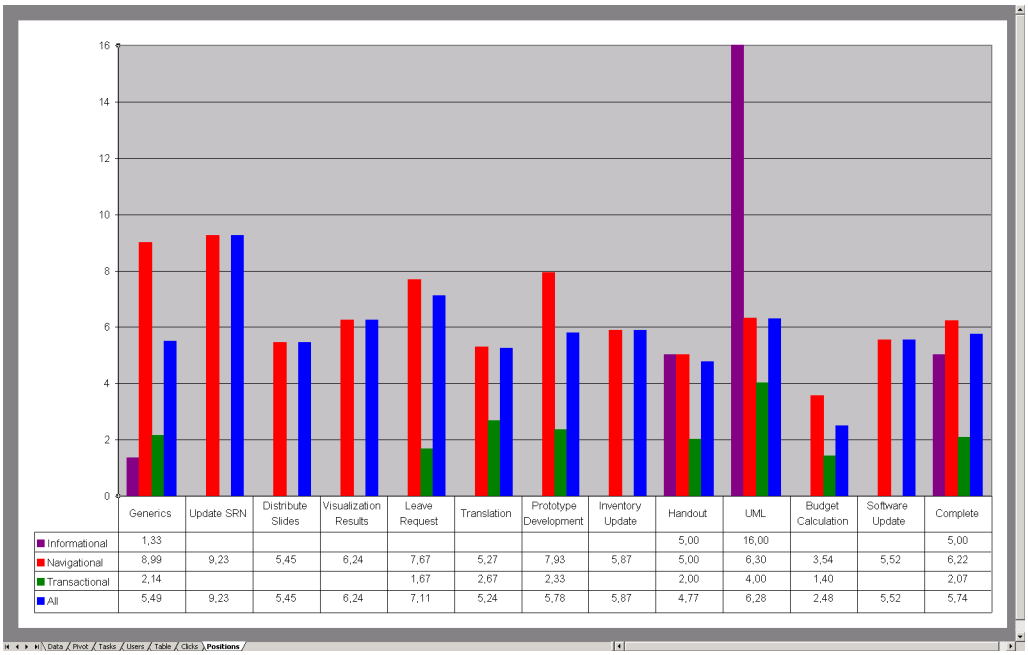
5.6 Questionnaire

To make this evaluation more comparable with the evaluations of other supporting systems we started our questionnaire with questions matching the DIN EN ISO 9241. Additionally we asked which of the three user goal



	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Developmen	Inventory Update	Handout	UML	Budget Calculation	Software Update
Informational Pseudo Clicked	0,20								0,27	0,07		
Navigational Pseudo Clicked	2,33	1,60	2,47	1,40	2,73	3,20	0,80	1,53	2,47	1,47	1,33	2,73
Transactional Pseudo Clicked	1,67	0,27	0,13		0,27	0,53	0,60	0,07	0,73	0,07	0,60	
All Pseudo Clicked	4,20	1,87	2,60	1,40	3,00	3,73	1,40	1,60	3,47	1,60	1,93	2,73
Informational Clicked	0,20								0,20	0,07		
Navigational Clicked	1,47	1,60	2,47	1,40	2,67	2,93	0,80	1,53	2,33	1,47	1,13	2,73
Transactional Clicked	1,20				0,20	0,33	0,60		0,13	0,07	0,53	
All Clicked	2,87	1,60	2,47	1,40	2,87	3,27	1,40	1,53	2,67	1,60	1,67	2,73

Figure 5.20: Field Study Results - Goal Support - Clicked Resource Counts



	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	Complete
Informational	1,33								5,00	16,00			5,00
Navigational	8,99	9,23	5,45	6,24	7,67	5,27	7,93	5,87	5,00	6,30	3,54	5,52	6,22
Transactional	2,14				1,67	2,67	2,33		2,00	4,00	1,40		2,07
All	5,49	9,23	5,45	6,24	7,11	5,24	5,78	5,87	4,77	6,28	2,48	5,52	5,74

Figure 5.21: Field Study Results - Goal Support - Resource Positions

supporting lists was most useful respectively less useful for the participants and during which task the recommended resources were good respectively bad in particular. Finally we asked them whether they could imagine to use a similar software in the future or not.

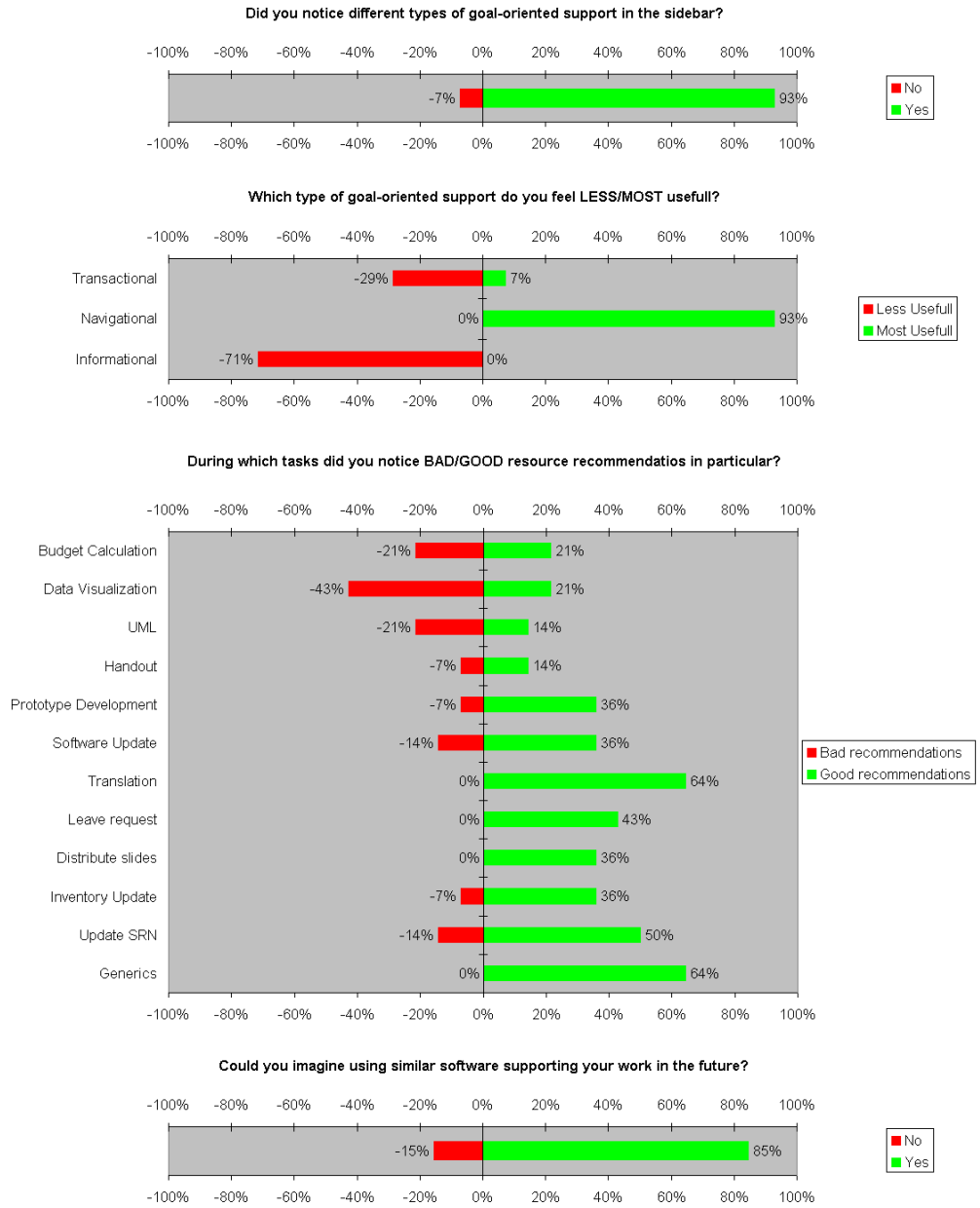


Figure 5.22: Questionaire Results - DIN Questions

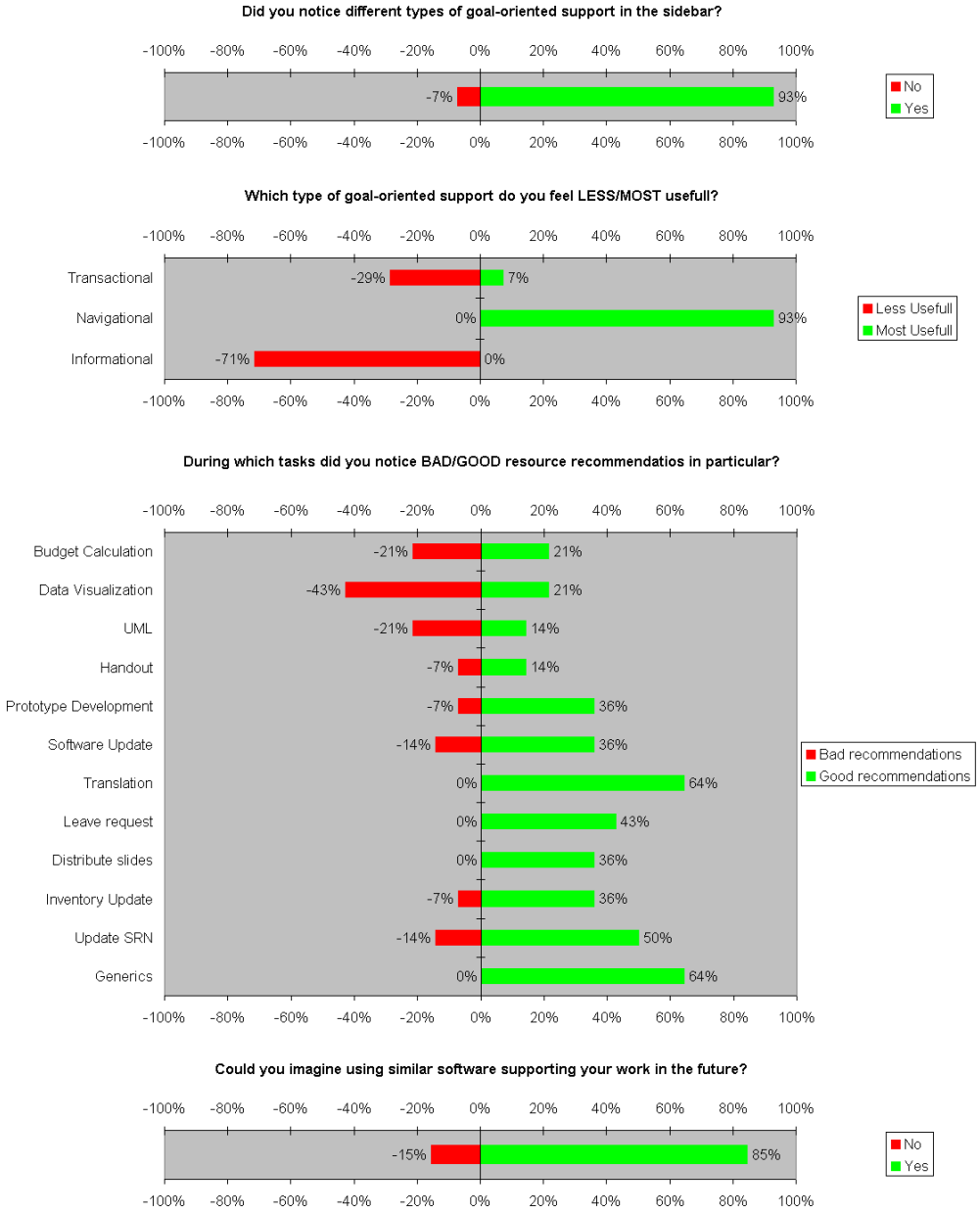


Figure 5.23: Questionaire Results - Goals And Tasks

Chapter 6

Conclusion

The initial intention of this thesis was to investigate how the different goals of a knowledge worker can be identified and supported. During the implementation and the evaluation we realized that it is hard to distinguish between the three different goals and that it is a good approach to support the *Navigational Goal*, the *Informational Goal*, and the *Transactional Goal* simultanously. The reason for this is that the different kinds of persons want to have their individual support for the different tasks they have to accomplish.

During the implementation of the algorithms and the evaluation platform we tried used a database and stored procedures to make the evaluation configurable and maintainable in a decentralized manner. By switching some entries in the database we could set the used algorithms and their configurations for all participants. Additionally we analysed if the algorithms run more efficient locally on the clients side oder as part of the database. The result were a clearly votum for the local algorithms and contra the ones running in the database.

With the help of our *User Interaction Emulator* and the collected data we could look back in the past and see what our participants would have seen when they would have been supported by our recommender algorithms. By this strategy we could debug and perfect our system. Since we collected very much logging data during the automatic analysis we could analyse the influance of many parameters on the algorithms perfomance. Not all of the results could be decribed in detail in the *Evaluation Chapter*.

During the field study the participants of our evaluation were very satisfied about the good recommendations of our alorithms. Especially the support for the navigational goal enjoed great popularity. The reason for the lower popularity of the support for the transactional goal could be the

small amount of learning data compared to the twelve partly to similar tasks. The spares use of the support for the informational goal can be justified by the selected tasks: They duration of a task was to short and there were to few tasks which needed a research for a theme or help to make something correctly.

Future Work

We analysed multiple aspects of the *Adaptive Support of Knowledge Work by Analysis of User Objectives*, but during our work we also found some aspects which could be analysed in a future research: For the navigational goal other algorithms could be investigated; maybe some of the graph algorithms could be modified for this purpose. To get better support for the transactional goal disambiguation of terms or using of position information could be analysed. Finally for the transactional goal the generation of new features could be interesting:They could extracted from some of the collected events or by combining multiple events to one more valuable feature. Additionally could the selection of other tasks with longer durations and other core themes lead to different results during a field study.

Bibliography

[AB08] MySQL AB, *Mysql 5.1 reference manual*, http://dev.mysql.com/doc/refman/5.1/en/index.html (2008).

[And88] J. R. Anderson, *A spreading activation theory of memory*, Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence (A. Collins and E. E. Smith, eds.), Kaufmann, San Mateo, CA, 1988, pp. 137–154.

[Bec06] Arne Beckhaus, *Machine learning on desktop enviroment events*, December 2006.

[Coo08] Microsoft Cooperation, *Named pipes*, http://msdn.microsoft.com/en-us/library/aa365590.aspx (2008).

[Fow97] Martin Fowler, *Uml distilled*, Addison-Wesley, 1997.

[Her94] Michael Herczeg, *Software-Ergonomie: Grundlagen der Mensch-Computer-Kommunikation*, Addison-Wesley, 1994.

[JBS07] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink, *Determining the user intent of web search engine queries*, WWW '07: Proceedings of the 16th international conference on World Wide Web (New York, NY, USA), ACM, 2007, pp. 1149–1150.

[KK] George Karypis and Vipin Kumar, *Metis: unstructured graph partitioning and sparse matrix ordering system*, Tech. report.

[LGFM08] Robert Lokaiczyk, Eicke Godehardt, Andreas Faatz, and Marek Meyer, *On resource acquisition in adaptive workplace-embedded e-learning environments*, Proceedings of the International Conference on E-Learning in the Workplace (ICELW), New York, USA, June 2008.

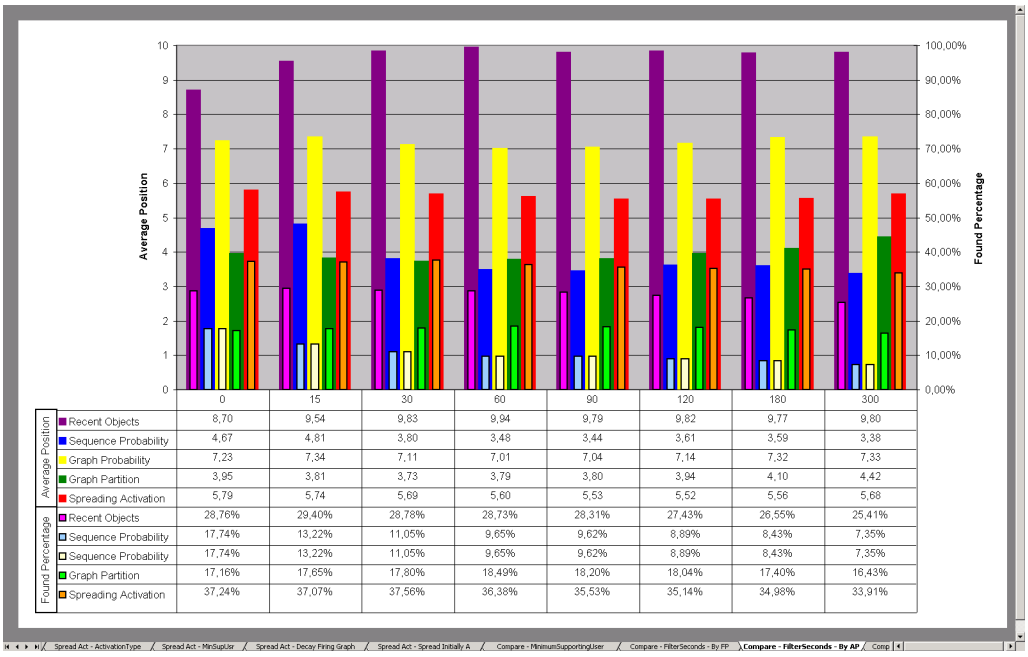
[Rip96] Brian D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

Appendix A

More experimental results

A.1 Further Analysis of Recommended Navigational Objects

A.1.1 Comparison



		0	15	30	60	90	120	180	300
Average Position	Recent Objects	8,70	9,54	9,83	9,94	9,79	9,82	9,77	9,80
	Sequence Probability	4,67	4,81	3,80	3,48	3,44	3,61	3,59	3,38
	Graph Probability	7,23	7,34	7,11	7,01	7,04	7,14	7,32	7,33
	Graph Partition	3,95	3,81	3,73	3,79	3,80	3,94	4,10	4,42
	Spreading Activation	5,79	5,74	5,69	5,60	5,53	5,52	5,56	5,68
Found Percentage	Recent Objects	28,76%	29,40%	28,78%	28,73%	28,31%	27,43%	26,55%	25,41%
	Sequence Probability	17,74%	13,22%	11,05%	9,65%	9,62%	8,89%	8,43%	7,35%
	Sequence Probability	17,74%	13,22%	11,05%	9,65%	9,62%	8,89%	8,43%	7,35%
	Graph Partition	17,16%	17,65%	17,80%	18,49%	18,20%	18,04%	17,40%	16,43%
	Spreading Activation	37,24%	37,07%	37,56%	36,38%	35,53%	35,14%	34,98%	33,91%

Figure A.1: Comparison of Navigational Recommenders - Influence of Filter Seconds - Best Average Positions

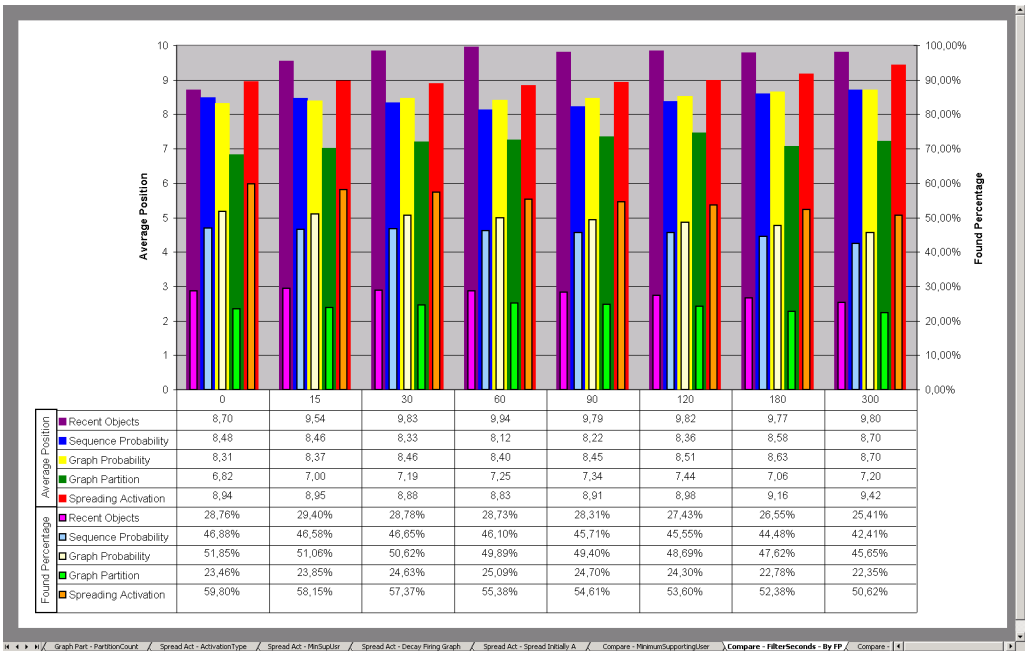


Figure A.2: Comparison of Navigational Recommenders - Influence of Filter Seconds - Best Found Percentages

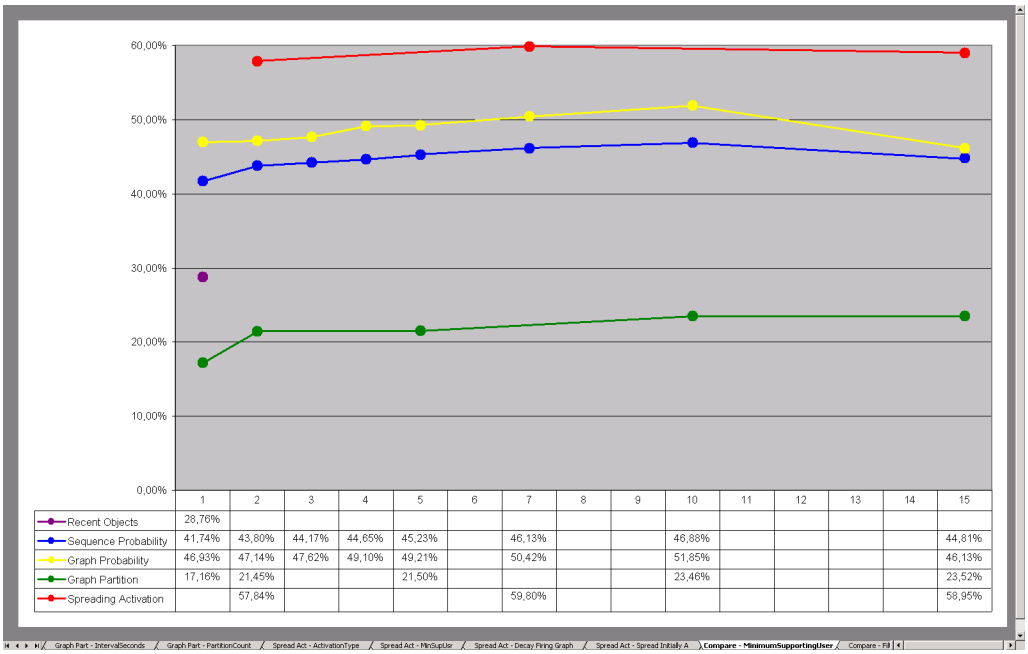
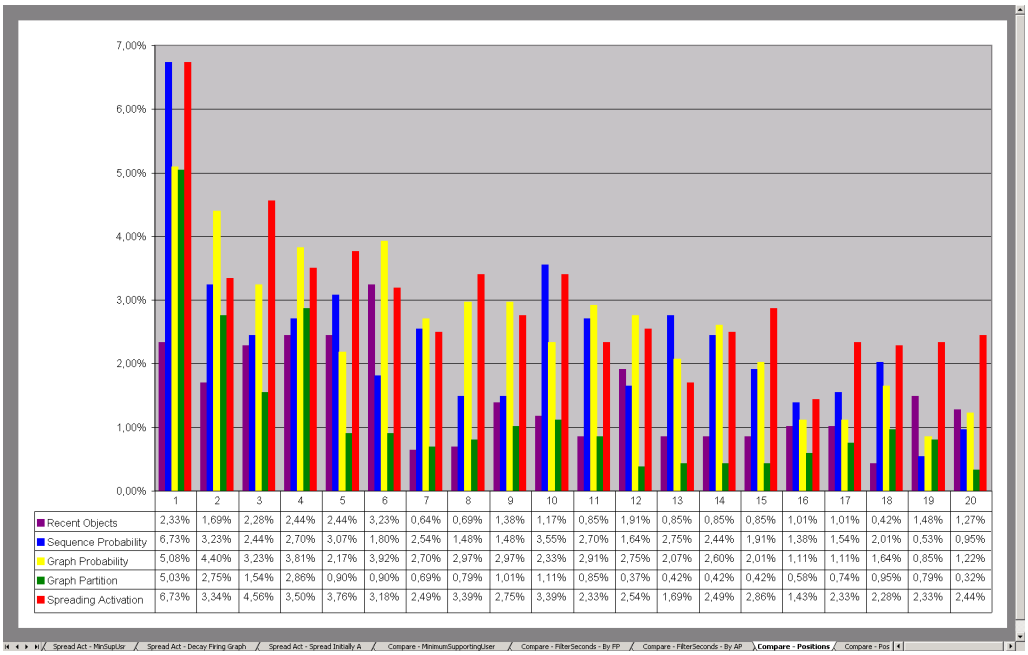


Figure A.3: Comparison of Navigational Recommenders - Influence of Minimum Supporting Users

A.1.2 Task Performances

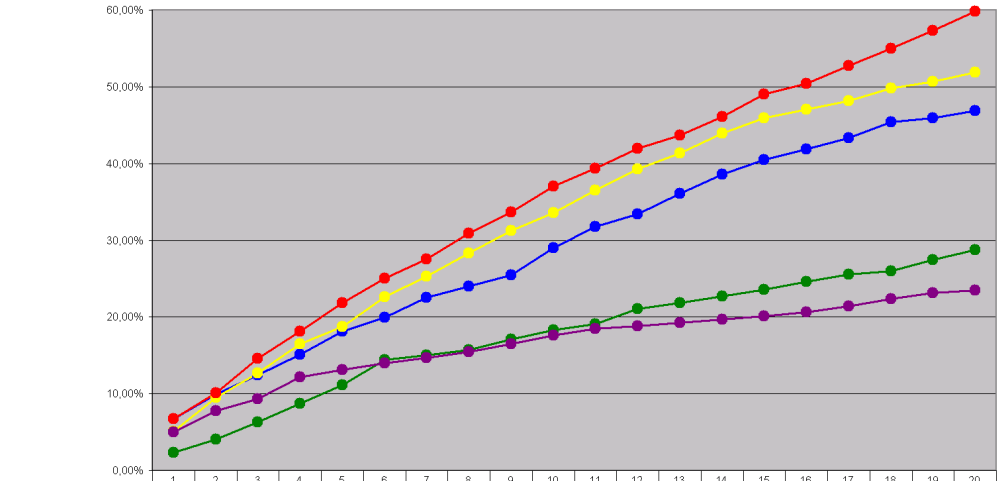
A.2 Analysis of Extracted Terms

For the algorithm supporting the Informational goal we realised that only the collected data of the task "*Create a one slide presentation on Generics*



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Recent Objects	2,33%	1,69%	2,28%	2,44%	2,44%	3,23%	0,64%	0,69%	1,38%	1,17%	0,85%	1,91%	0,85%	0,85%	0,85%	1,01%	1,01%	0,42%	1,48%	1,27%
Sequence Probability	6,73%	3,23%	2,44%	2,70%	3,07%	1,80%	2,54%	1,48%	1,48%	3,55%	2,70%	1,64%	2,75%	2,44%	1,91%	1,38%	1,54%	2,01%	0,53%	0,95%
Graph Probability	5,08%	4,40%	3,23%	3,81%	2,17%	3,92%	2,70%	2,97%	2,97%	2,33%	2,91%	2,75%	2,07%	2,60%	2,01%	1,11%	1,11%	1,64%	0,85%	1,22%
Graph Partition	5,03%	2,75%	1,54%	2,86%	0,90%	0,90%	0,69%	0,79%	1,01%	1,11%	0,85%	0,37%	0,42%	0,42%	0,42%	0,58%	0,74%	0,95%	0,79%	0,32%
Spreading Activation	6,73%	3,34%	4,56%	3,50%	3,76%	3,18%	2,49%	3,39%	2,75%	3,39%	2,33%	2,54%	1,69%	2,49%	2,86%	1,43%	2,33%	2,28%	2,33%	2,44%

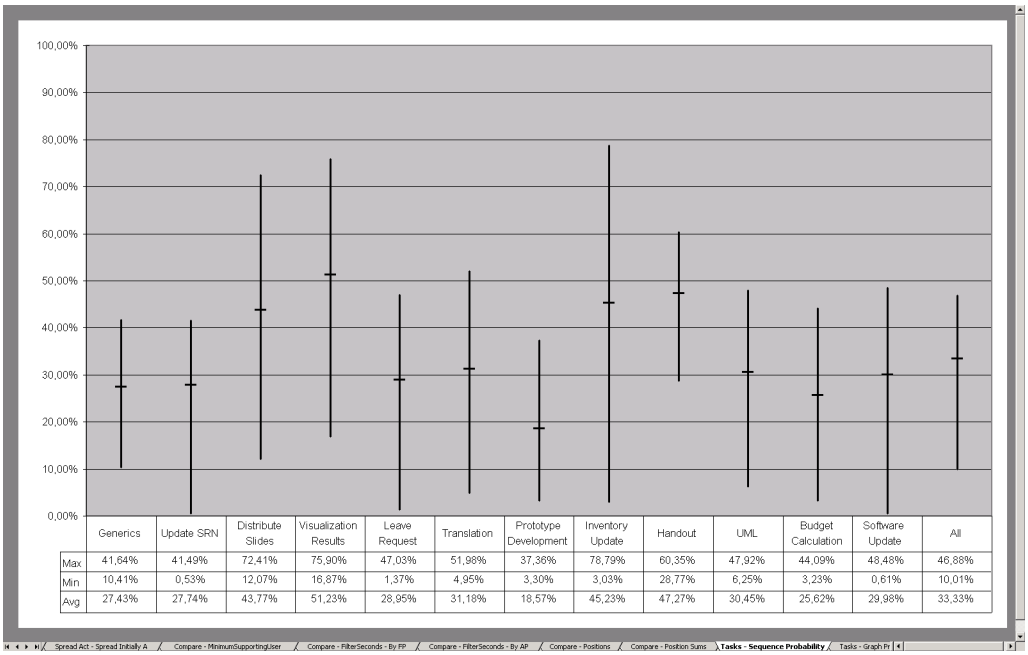
Figure A.4: Comparison of Navigational Recommenders - Found Percentages on Positions



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Recent Objects	2,33%	4,03%	6,30%	8,74%	11,18%	14,41%	15,04%	15,73%	17,11%	18,27%	19,12%	21,03%	21,87%	22,72%	23,57%	24,58%	25,58%	26,01%	27,49%	28,76%
Sequence Probability	6,73%	9,96%	12,39%	15,10%	18,17%	19,97%	22,51%	23,99%	25,48%	29,03%	31,73%	33,37%	36,12%	38,56%	40,47%	41,84%	43,38%	45,39%	45,92%	46,87%
Graph Probability	5,08%	9,48%	12,71%	16,53%	18,70%	22,62%	25,32%	28,28%	31,25%	33,58%	36,49%	39,25%	41,31%	43,91%	45,92%	47,03%	48,15%	49,79%	50,64%	51,85%
Graph Partition	5,03%	7,79%	9,32%	12,18%	13,08%	13,98%	14,67%	15,47%	16,47%	17,58%	18,43%	18,80%	19,23%	19,65%	20,07%	20,66%	21,40%	22,35%	23,15%	23,46%
Spreading Activation	6,73%	10,06%	14,62%	18,11%	21,87%	25,05%	27,54%	30,93%	33,69%	37,08%	39,41%	41,95%	43,64%	46,13%	48,99%	50,42%	52,75%	55,03%	57,36%	59,80%

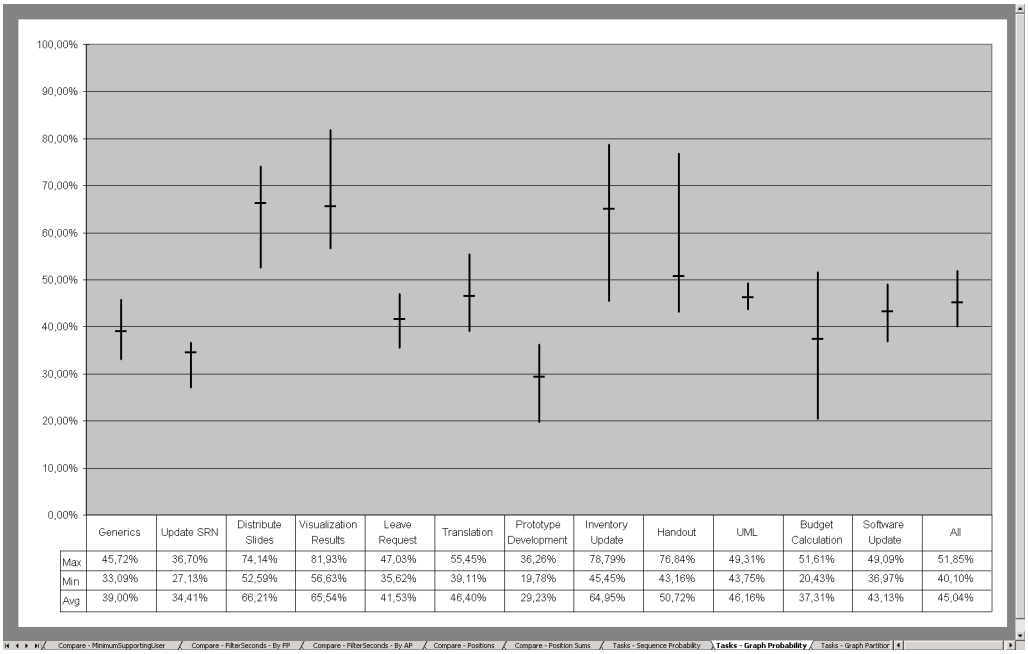
Figure A.5: Comparison of Navigational Recommenders - Accumulated Found Percentages on Positions

in Java" provided us with compareble results. The other tasks were to short and did not need a support for the informational goal. We analysed how often the term "*Java*" was extracted and on which averaged position in was in the extracted term list.



	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	All
Max	41,64%	41,49%	72,41%	75,90%	47,03%	51,98%	37,36%	78,79%	60,35%	47,92%	44,09%	48,48%	46,88%
Min	10,41%	0,53%	12,07%	16,87%	1,37%	4,95%	3,30%	3,03%	28,77%	6,25%	3,23%	0,61%	10,01%
Avg	27,43%	27,74%	43,77%	51,23%	28,95%	31,18%	18,57%	45,23%	47,27%	30,45%	25,62%	29,98%	33,33%

Figure A.6: Tasks Performances of Navigational Recommenders - Sequence Probability Recommender

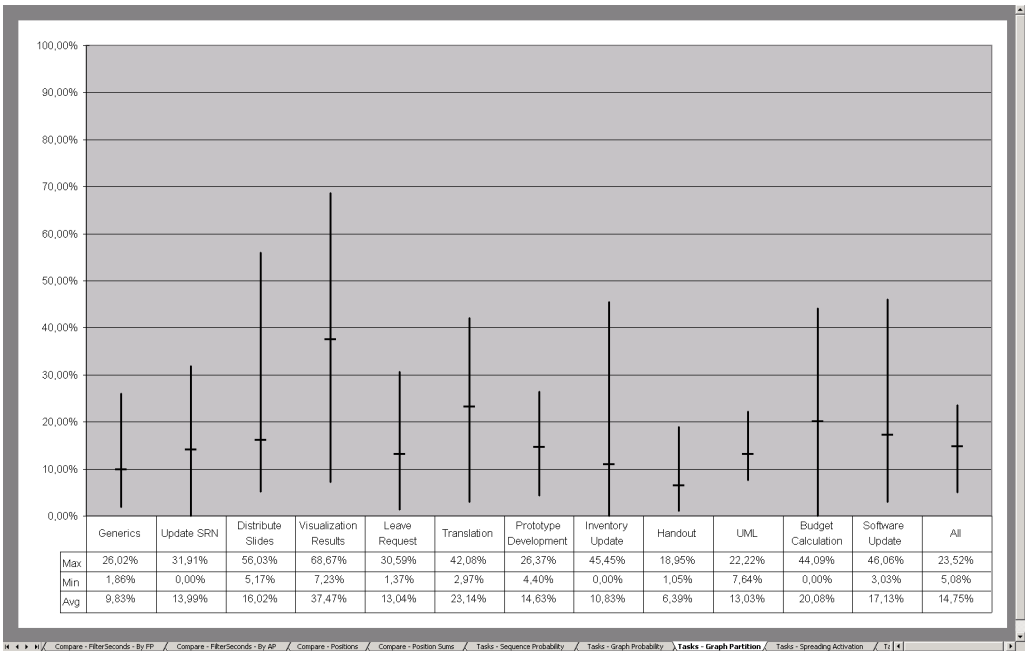


	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	All
Max	45,72%	36,70%	74,14%	81,93%	47,03%	55,45%	36,26%	78,79%	76,84%	49,31%	51,61%	49,09%	51,85%
Min	33,09%	27,13%	52,59%	56,63%	35,62%	39,11%	19,78%	45,45%	43,16%	43,75%	20,43%	36,97%	40,10%
Avg	39,00%	34,41%	66,21%	65,54%	41,53%	46,40%	29,23%	64,95%	50,72%	46,16%	37,31%	43,13%	45,04%

Figure A.7: Tasks Performances of Navigational Recommenders - Graph Probability Recommender

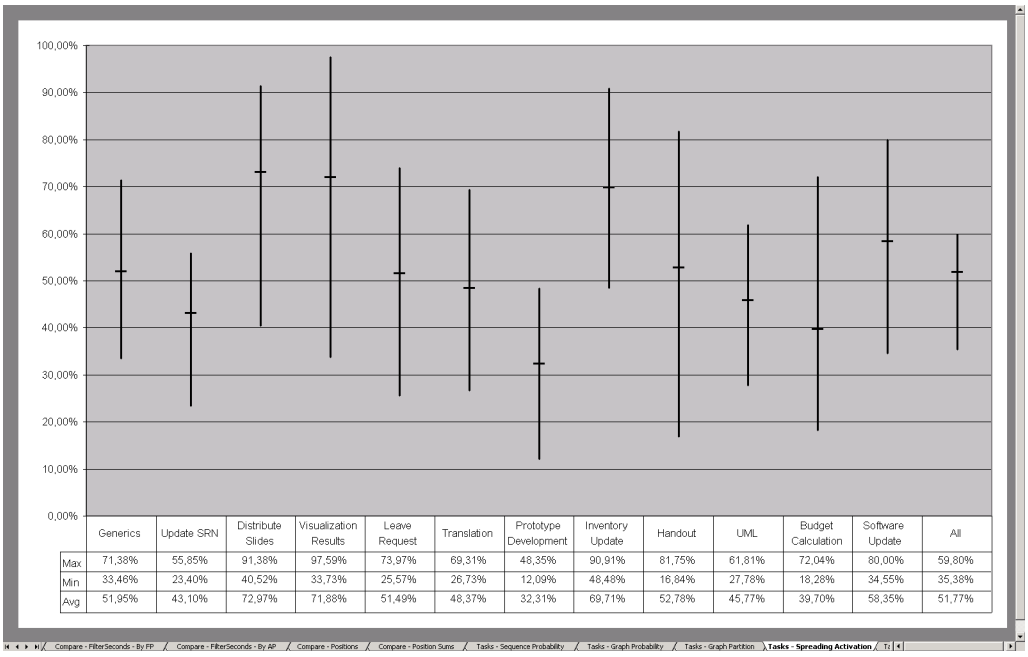
For the Term Extraction Algorithm we analysed the influance of the following parameters:

- *Used Event Types*



	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	All
Max	26,02%	31,91%	56,03%	68,67%	30,59%	42,08%	26,37%	45,45%	18,95%	22,22%	44,09%	46,06%	23,52%
Min	1,86%	0,00%	5,17%	7,23%	1,37%	2,97%	4,40%	0,00%	1,05%	7,64%	0,00%	3,03%	5,08%
Avg	9,83%	13,99%	16,02%	37,47%	13,04%	23,14%	14,63%	10,83%	6,39%	13,03%	20,08%	17,13%	14,75%

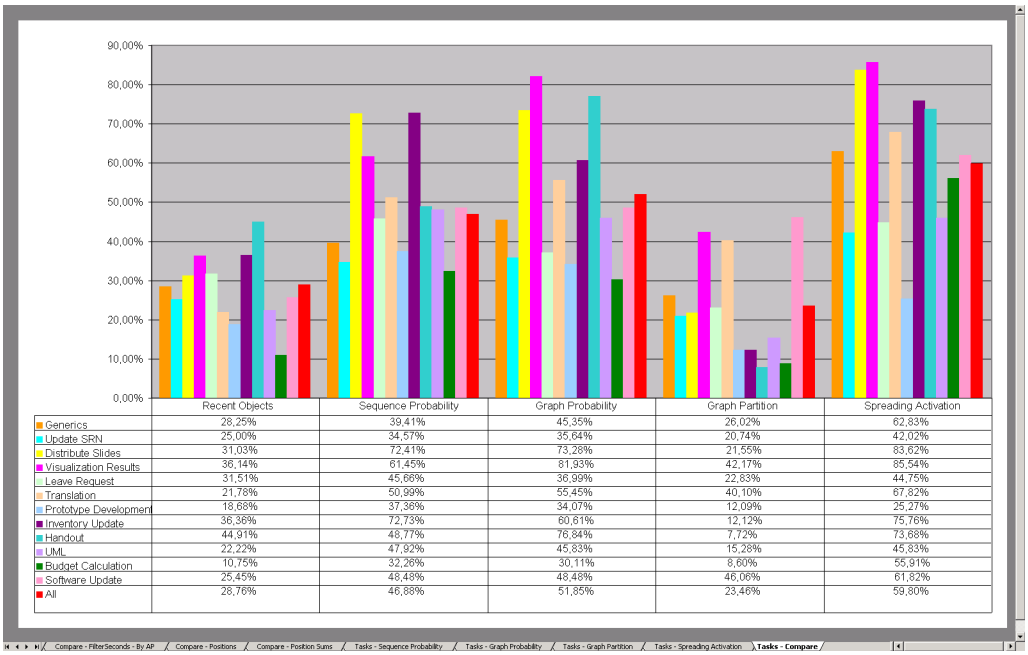
Figure A.8: Tasks Performances of Navigational Recommenders - Graph Partition Recommender



	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	All
Max	71,38%	55,85%	91,38%	97,59%	73,97%	69,31%	48,35%	90,91%	81,75%	61,81%	72,04%	80,00%	59,80%
Min	33,46%	23,40%	40,52%	33,73%	25,57%	26,73%	12,09%	48,48%	16,84%	27,78%	18,28%	34,55%	35,38%
Avg	51,95%	43,10%	72,97%	71,88%	51,49%	48,37%	32,31%	69,71%	52,78%	45,77%	39,70%	58,35%	51,77%

Figure A.9: Tasks Performances of Navigational Recommenders - Spreading Activation Recommender

- *Keyboard Events*
- *Clipboard Events*
- *Application Events*



	Recent Objects	Sequence Probability	Graph Probability	Graph Partition	Spreading Activation
Generics	28,25%	39,41%	45,35%	26,02%	62,83%
Update SRN	25,00%	34,57%	35,64%	20,74%	42,02%
Distribute Slides	31,03%	72,41%	73,28%	21,55%	83,62%
Visualization Results	36,14%	61,45%	81,93%	42,17%	85,54%
Leave Request	31,51%	45,66%	36,99%	22,83%	44,75%
Translation	21,78%	50,99%	55,45%	40,10%	67,82%
Prototype Development	18,68%	37,36%	34,07%	12,09%	25,27%
Inventory Update	36,36%	72,73%	60,61%	12,12%	75,76%
Handout	44,91%	48,77%	76,84%	7,72%	73,68%
UML	22,22%	47,92%	45,83%	15,28%	45,83%
Budget Calculation	10,75%	32,26%	30,11%	8,60%	55,91%
Software Update	25,45%	48,48%	48,48%	46,06%	61,82%
All	28,76%	46,88%	51,85%	23,46%	59,80%

Figure A.10: Tasks Performances of Navigational Recommenders - Comparison

  - *Foreground Windows Events*
  - *Combinations* - Keyboard/Clipboard, Keyboard/Clipboard/Application, all four

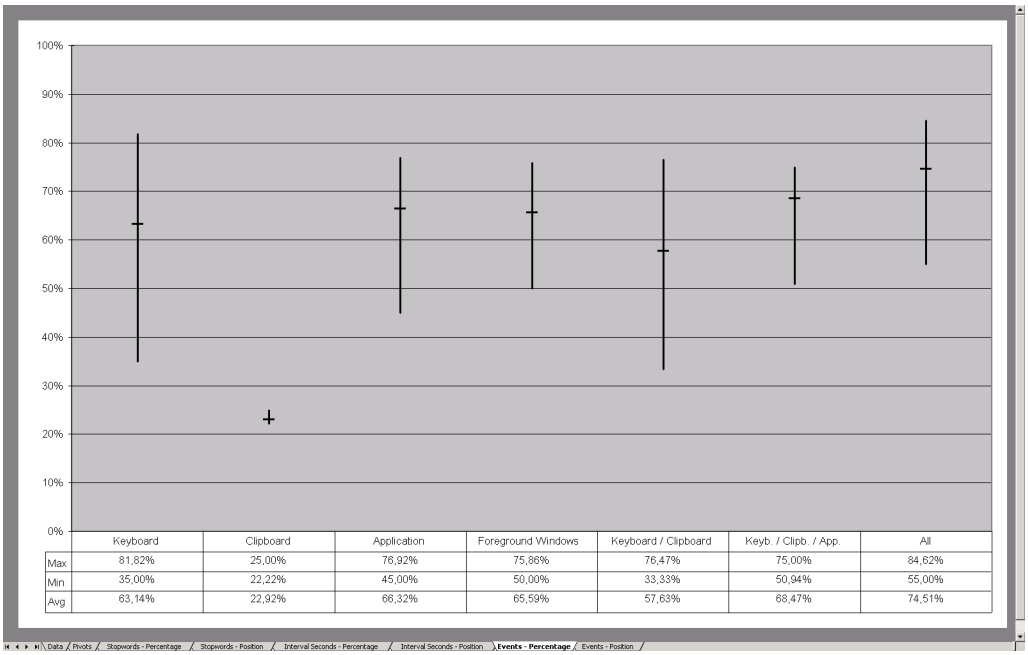
- *Extraction Interval Seconds* - Values: 30, 60, 90, 120
- *Stopword Count* - Values: 100, 1.000, 10.000
- *Stopword Language* - English, German, or mixed English-German

A.3 Analysis of Classified Tasks

A.3.1 Comparison of Classifier Algorithms

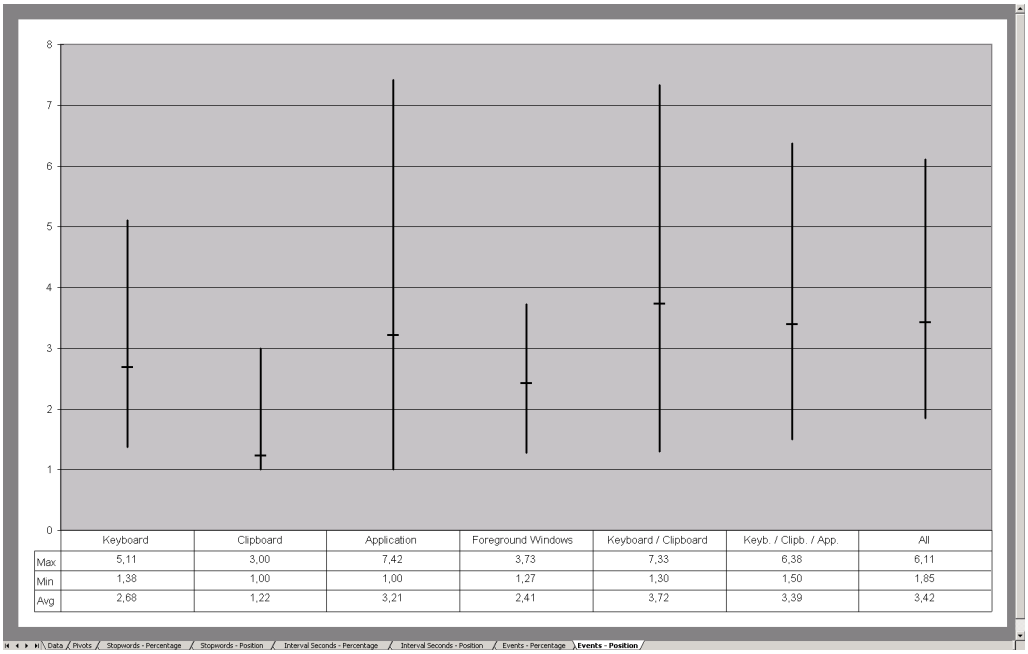
A.3.2 Sequential Minimal Optimization Classifier

A.3.3 Voting Classifier



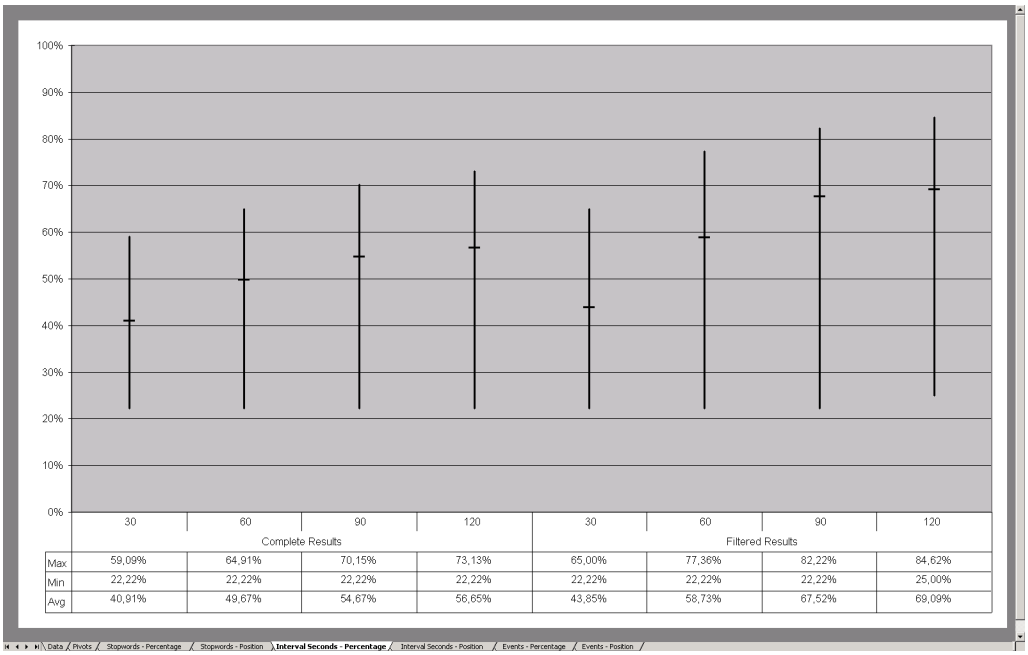
	Keyboard	Clipboard	Application	Foreground Windows	Keyboard / Clipboard	Keyb. / Clipb. / App.	All
Max	81,82%	25,00%	76,92%	75,86%	76,47%	75,00%	84,62%
Min	35,00%	22,22%	45,00%	50,00%	33,33%	50,94%	55,00%
Avg	63,14%	22,92%	66,32%	65,59%	57,63%	68,47%	74,51%

Figure A.11: Informational Recommender - Influence of Used Event Types - Found Percentages



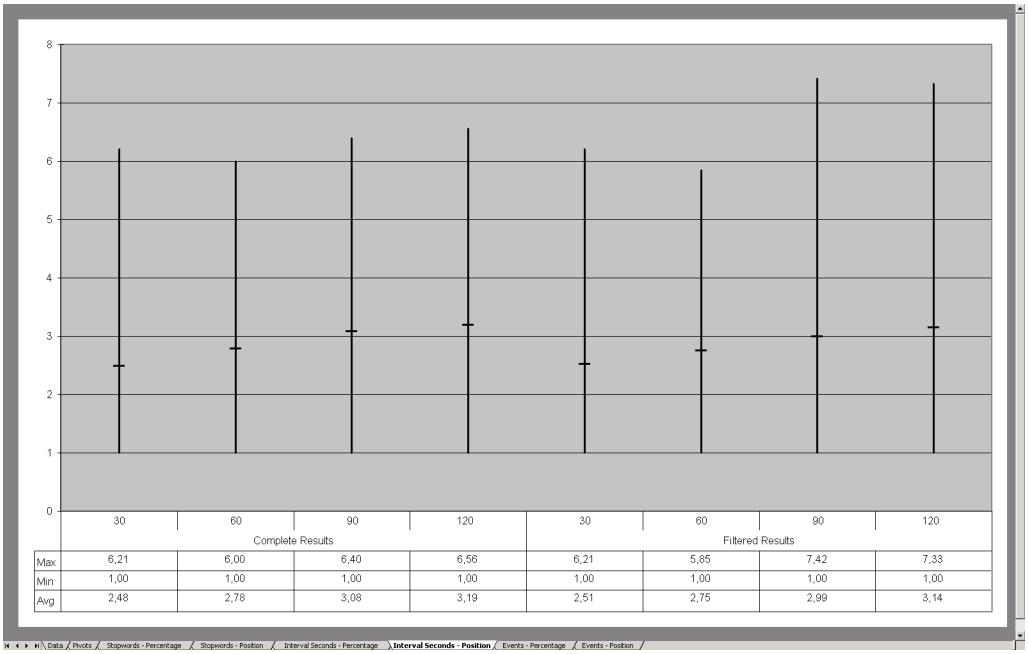
	Keyboard	Clipboard	Application	Foreground Windows	Keyboard / Clipboard	Keyb. / Clipb. / App.	All
Max	5,11	3,00	7,42	3,73	7,33	6,38	6,11
Min	1,38	1,00	1,00	1,27	1,30	1,50	1,85
Avg	2,68	1,22	3,21	2,41	3,72	3,39	3,42

Figure A.12: Informational Recommender - Influence of Used Event Types - Average Positions



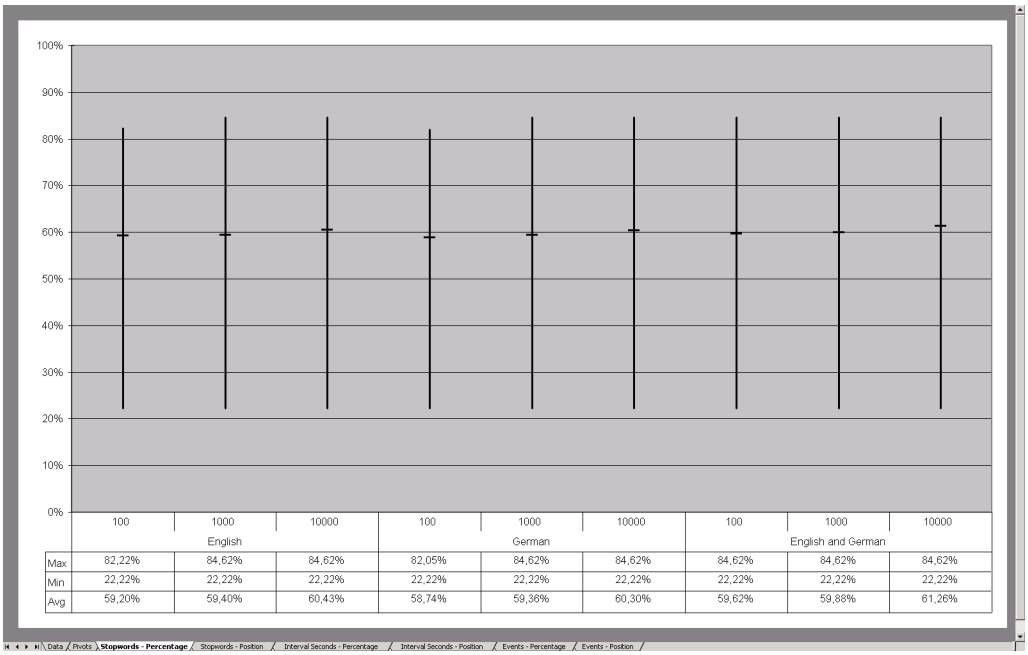
	30	60	90	120	30	60	90	120
	Complete Results				Filtered Results			
Max	59,09%	64,91%	70,15%	73,13%	65,00%	77,38%	82,22%	84,62%
Min	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%	25,00%
Avg	40,91%	49,67%	54,67%	56,65%	43,85%	58,73%	67,52%	69,09%

Figure A.13: Informational Recommender - Influence of Interval Seconds - Found Percentages



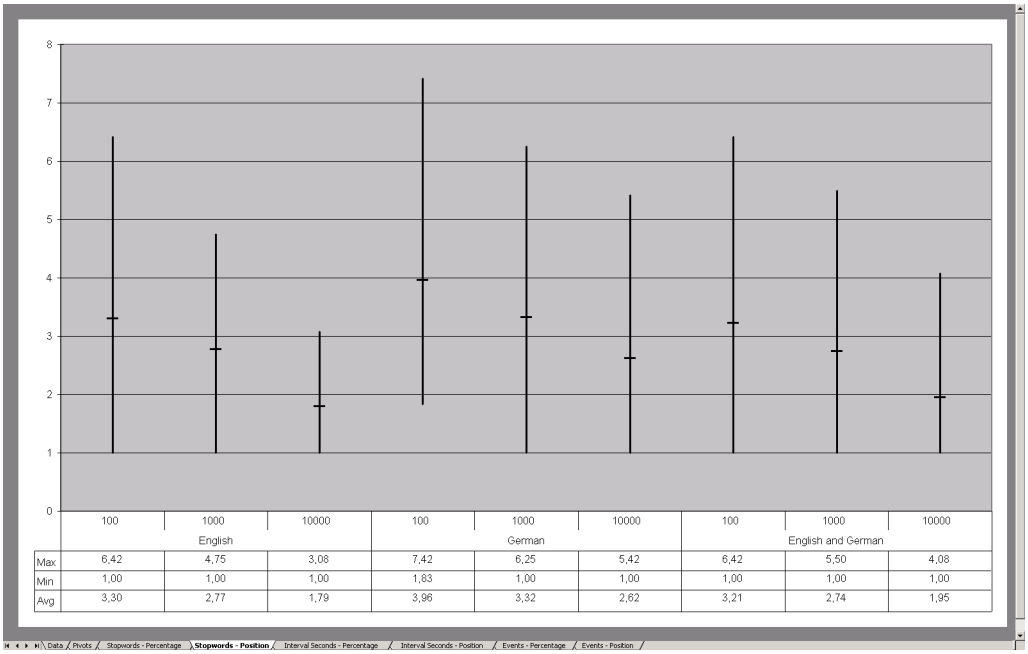
	30	60	90	120	30	60	90	120
	Complete Results				Filtered Results			
Max	6,21	6,00	6,40	6,56	6,21	5,85	7,42	7,33
Min	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Avg	2,48	2,78	3,08	3,19	2,51	2,75	2,99	3,14

Figure A.14: Informational Recommender - Influence of Interval Seconds - Average Positions



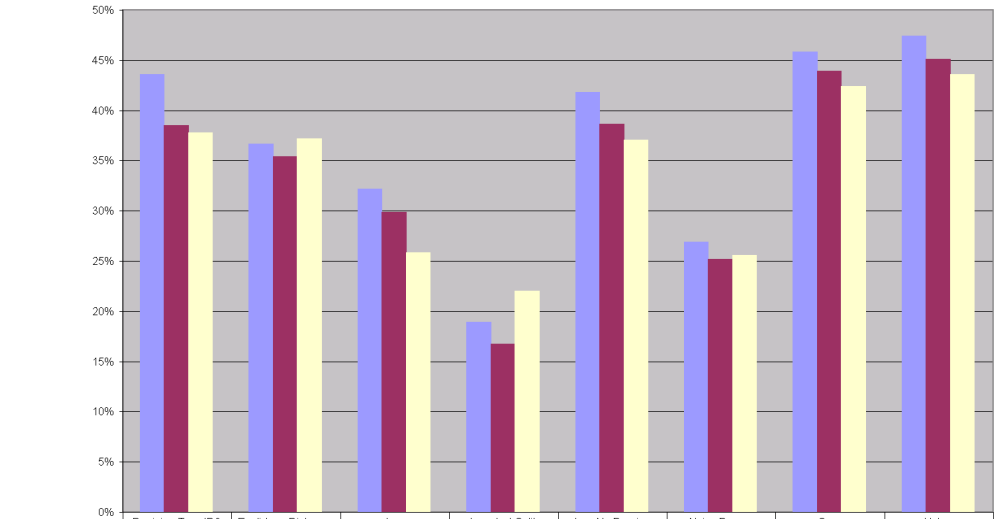
	100	1000	10000	100	1000	10000	100	1000	10000
	English			German			English and German		
Max	82,22%	84,62%	84,62%	82,05%	84,62%	84,62%	84,62%	84,62%	84,62%
Min	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%	22,22%
Avg	59,20%	59,40%	60,43%	58,74%	59,36%	60,30%	59,62%	59,88%	61,26%

Figure A.15: Informational Recommender - Influence of Used Stopwords - Found Percentages



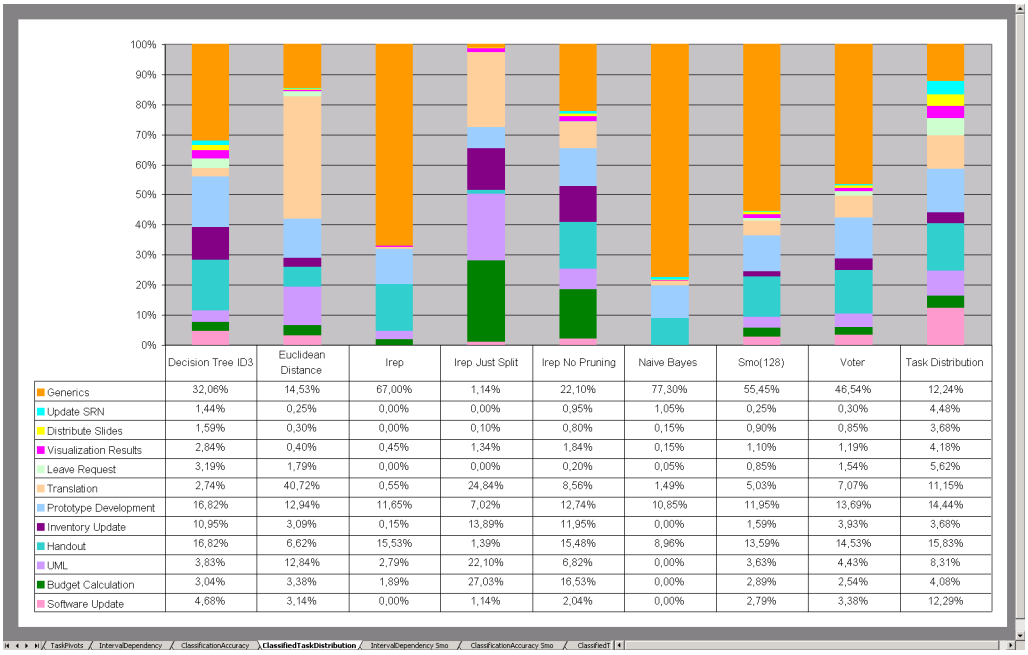
	100	1000	10000	100	1000	10000	100	1000	10000
	English			German			English and German		
Max	6,42	4,75	3,08	7,42	6,25	5,42	6,42	5,50	4,08
Min	1,00	1,00	1,00	1,83	1,00	1,00	1,00	1,00	1,00
Avg	3,30	2,77	1,79	3,96	3,32	2,62	3,21	2,74	1,95

Figure A.16: Informational Recommender - Influence of Used Stopwords - Average Positions



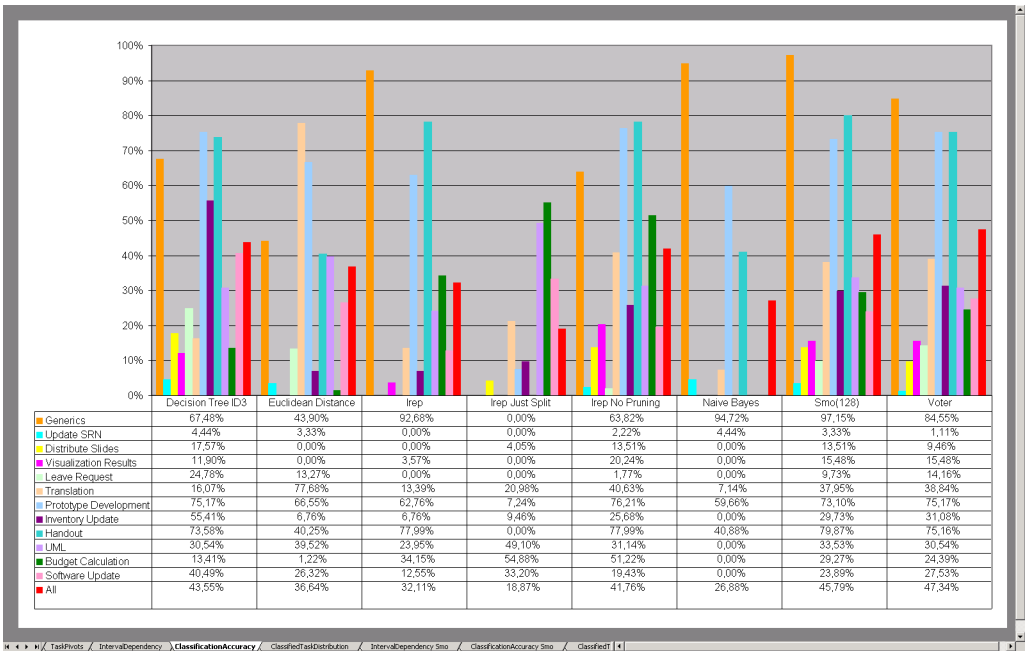
	Decision Tree ID3	Euclidean Distance	Irep	Irep Just Split	Irep No Pruning	Naive Bayes	Smo	Voter
Interval Seconds 5	43,55%	36,64%	32,11%	18,87%	41,76%	26,88%	45,79%	47,34%
Interval Seconds 10	38,45%	35,33%	29,82%	16,71%	38,61%	25,10%	43,88%	45,08%
Interval Seconds 15	37,76%	37,11%	25,79%	21,97%	36,97%	25,53%	42,37%	43,55%

Figure A.17: Transactional Recommender - Classifier Comparison - Influence of Interval Seconds



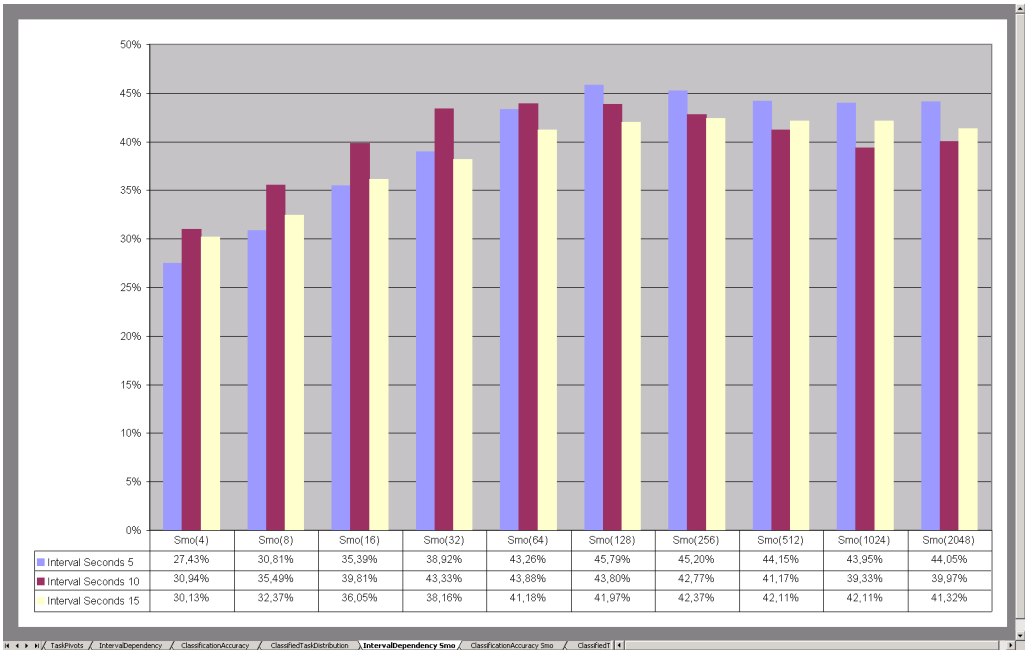
	Decision Tree ID3	Euclidean Distance	Irep	Irep Just Split	Irep No Pruning	Naive Bayes	Smo(128)	Voter	Task Distribution
Generics	32,06%	14,53%	67,00%	1,14%	22,10%	77,30%	55,45%	46,54%	12,24%
Update SRN	1,44%	0,25%	0,00%	0,00%	0,95%	1,05%	0,25%	0,30%	4,48%
Distribute Slides	1,59%	0,30%	0,00%	0,10%	0,80%	0,15%	0,90%	0,85%	3,68%
Visualization Results	2,84%	0,40%	0,45%	1,34%	1,84%	0,15%	1,10%	1,19%	4,18%
Leave Request	3,19%	1,79%	0,00%	0,00%	0,20%	0,05%	0,85%	1,54%	5,62%
Translation	2,74%	40,72%	0,55%	24,84%	8,56%	1,49%	5,03%	7,07%	11,15%
Prototype Development	16,82%	12,94%	11,65%	7,02%	12,74%	10,85%	11,95%	13,69%	14,44%
Inventory Update	10,95%	3,09%	0,15%	13,89%	11,95%	0,00%	1,59%	3,93%	3,68%
Handout	16,82%	6,62%	15,53%	1,39%	15,48%	8,96%	13,59%	14,53%	15,83%
UML	3,83%	12,84%	2,79%	22,10%	6,82%	0,00%	3,63%	4,43%	8,31%
Budget Calculation	3,04%	3,38%	1,89%	27,03%	16,53%	0,00%	2,89%	2,54%	4,08%
Software Update	4,68%	3,14%	0,00%	1,14%	2,04%	0,00%	2,79%	3,38%	12,29%

Figure A.18: Transactional Recommender - Classifier Comparison - Classified Task Distribution



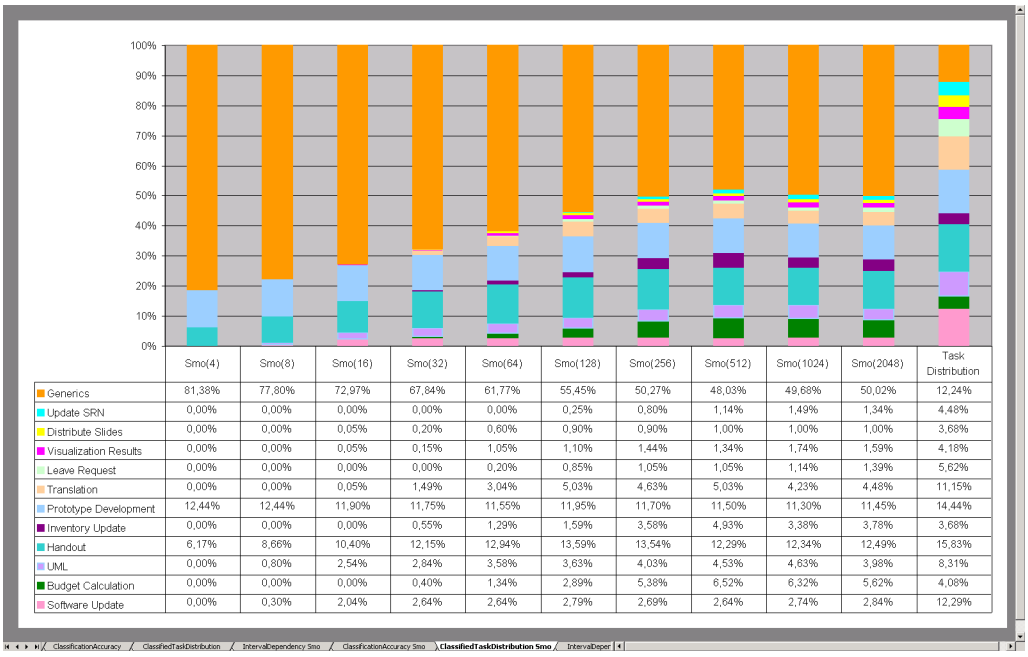
	Decision Tree ID3	Euclidean Distance	Irep	Irep Just Split	Irep No Pruning	Naive Bayes	Smo(128)	Voter
Generics	67,48%	43,90%	92,68%	0,00%	63,82%	94,72%	97,15%	84,55%
Update SRN	4,44%	3,33%	0,00%	0,00%	2,22%	4,44%	3,33%	1,11%
Distribute Slides	17,57%	0,00%	0,00%	4,05%	13,51%	0,00%	13,51%	9,46%
Visualization Results	11,90%	0,00%	3,57%	0,00%	20,24%	0,00%	15,48%	15,48%
Leave Request	24,78%	13,27%	0,00%	0,00%	1,77%	0,00%	9,73%	14,16%
Translation	16,07%	77,68%	13,39%	20,98%	40,63%	7,14%	37,95%	38,84%
Prototype Development	75,17%	66,55%	62,76%	7,24%	76,21%	59,66%	73,10%	75,17%
Inventory Update	55,41%	6,76%	6,76%	9,46%	25,68%	0,00%	29,73%	31,08%
Handout	73,58%	40,25%	77,99%	0,00%	77,99%	40,88%	79,87%	75,16%
UML	30,54%	39,52%	23,95%	49,10%	31,14%	0,00%	33,53%	30,54%
Budget Calculation	13,41%	1,22%	34,15%	54,88%	51,22%	0,00%	29,27%	24,39%
Software Update	40,49%	26,32%	12,55%	33,20%	19,43%	0,00%	23,89%	27,53%
All	43,55%	36,64%	32,11%	18,87%	41,76%	26,88%	45,79%	47,34%

Figure A.19: Transactional Recommender - Classifier Comparison - Classification Accuracy



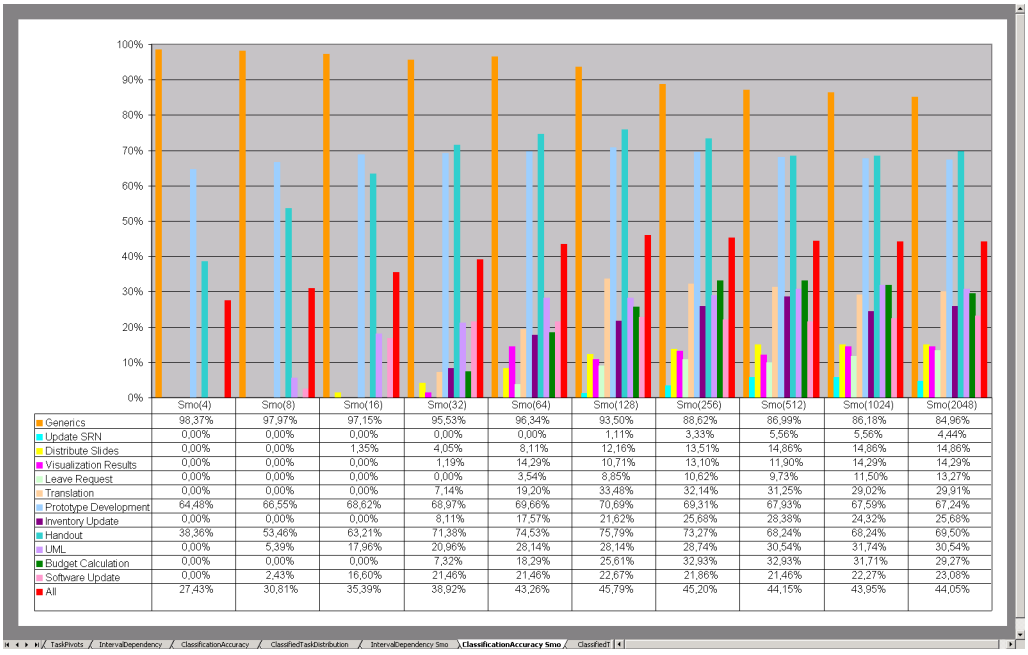
	Smo(4)	Smo(8)	Smo(16)	Smo(32)	Smo(64)	Smo(128)	Smo(256)	Smo(512)	Smo(1024)	Smo(2048)
Interval Seconds 5	27,43%	30,81%	35,39%	38,92%	43,26%	45,79%	45,20%	44,15%	43,95%	44,05%
Interval Seconds 10	30,94%	35,49%	39,81%	43,33%	43,88%	43,80%	42,77%	41,17%	39,33%	39,97%
Interval Seconds 15	30,13%	32,37%	36,05%	38,16%	41,18%	41,97%	42,37%	42,11%	42,11%	41,32%

Figure A.20: Transactional Recommender - SMO Classifier - Influence of Interval Seconds



	Smo(4)	Smo(8)	Smo(16)	Smo(32)	Smo(64)	Smo(128)	Smo(256)	Smo(512)	Smo(1024)	Smo(2048)	Task Distribution
Generics	81,38%	77,80%	72,97%	67,84%	61,77%	55,45%	50,27%	48,03%	49,68%	50,02%	12,24%
Update SRN	0,00%	0,00%	0,00%	0,00%	0,00%	0,25%	0,80%	1,14%	1,49%	1,34%	4,48%
Distribute Slides	0,00%	0,00%	0,05%	0,20%	0,60%	0,90%	0,90%	1,00%	1,00%	1,00%	3,68%
Visualization Results	0,00%	0,00%	0,05%	0,15%	1,05%	1,10%	1,44%	1,34%	1,74%	1,59%	4,18%
Leave Request	0,00%	0,00%	0,00%	0,00%	0,20%	0,85%	1,05%	1,05%	1,14%	1,39%	5,62%
Translation	0,00%	0,00%	0,05%	1,49%	3,04%	5,03%	4,63%	5,03%	4,23%	4,48%	11,15%
Prototype Development	12,44%	12,44%	11,90%	11,75%	11,55%	11,95%	11,70%	11,50%	11,30%	11,45%	14,44%
Inventory Update	0,00%	0,00%	0,00%	0,55%	1,29%	1,59%	3,58%	4,93%	3,38%	3,78%	3,68%
Handout	6,17%	8,66%	10,40%	12,15%	12,94%	13,59%	13,54%	12,29%	12,34%	12,49%	15,83%
UML	0,00%	0,80%	2,54%	2,84%	3,58%	3,63%	4,03%	4,53%	4,63%	3,98%	8,31%
Budget Calculation	0,00%	0,00%	0,00%	0,40%	1,34%	2,89%	5,38%	6,52%	6,32%	5,62%	4,08%
Software Update	0,00%	0,30%	2,04%	2,64%	2,64%	2,79%	2,69%	2,64%	2,74%	2,84%	12,29%

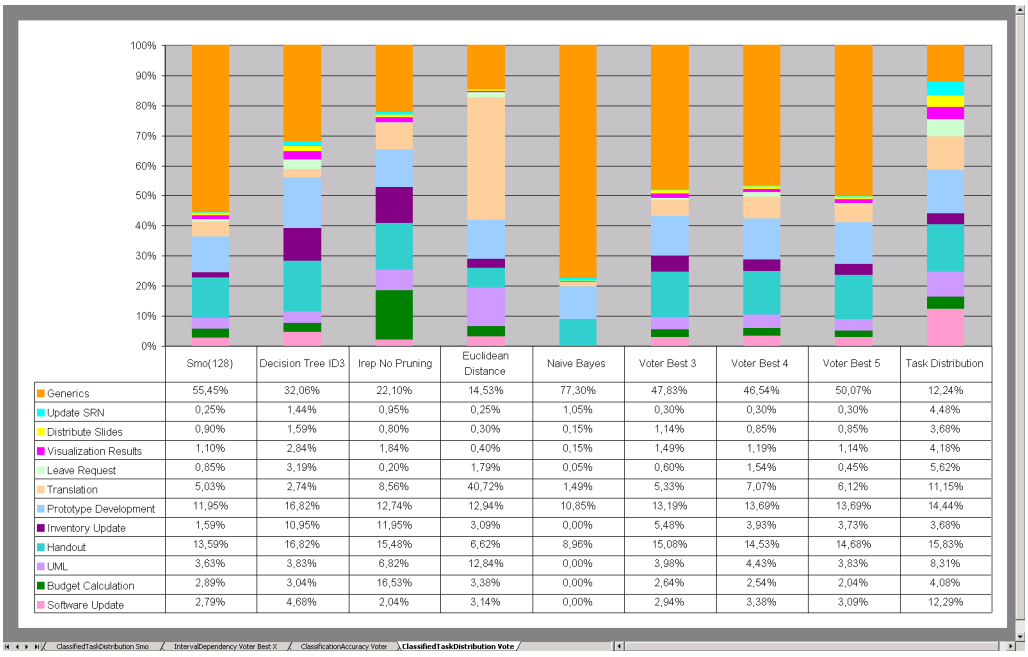
Figure A.21: Transactional Recommender - SMO Classifier - Classified Task Distribution



	Smo(4)	Smo(8)	Smo(16)	Smo(32)	Smo(64)	Smo(128)	Smo(256)	Smo(512)	Smo(1024)	Smo(2048)
Generics	98,37%	97,97%	97,15%	95,53%	96,34%	93,50%	88,62%	86,99%	86,18%	84,96%
Update SRN	0,00%	0,00%	0,00%	0,00%	0,00%	1,11%	3,33%	5,56%	5,56%	4,44%
Distribute Slides	0,00%	0,00%	1,35%	4,05%	8,11%	12,16%	13,51%	14,86%	14,86%	14,86%
Visualization Results	0,00%	0,00%	0,00%	1,19%	14,29%	10,71%	13,10%	11,90%	14,29%	14,29%
Leave Request	0,00%	0,00%	0,00%	0,00%	3,54%	8,85%	10,62%	9,73%	11,50%	13,27%
Translation	0,00%	0,00%	0,00%	7,14%	19,20%	33,48%	32,14%	31,25%	29,02%	29,91%
Prototype Development	64,48%	66,55%	68,62%	68,97%	69,66%	70,69%	69,31%	67,93%	67,59%	67,24%
Inventory Update	0,00%	0,00%	0,00%	8,11%	17,57%	21,62%	25,68%	28,38%	24,32%	25,68%
Handout	38,36%	53,46%	63,21%	71,38%	74,53%	75,79%	73,27%	68,24%	68,24%	69,50%
UML	0,00%	5,39%	17,96%	20,96%	28,14%	28,14%	28,74%	30,54%	31,74%	30,54%
Budget Calculation	0,00%	0,00%	0,00%	7,32%	18,29%	25,61%	32,93%	32,93%	31,71%	29,27%
Software Update	0,00%	2,43%	16,60%	21,46%	21,46%	22,67%	21,86%	21,46%	22,27%	23,08%
All	27,43%	30,81%	35,39%	38,92%	43,26%	45,79%	45,20%	44,15%	43,95%	44,05%

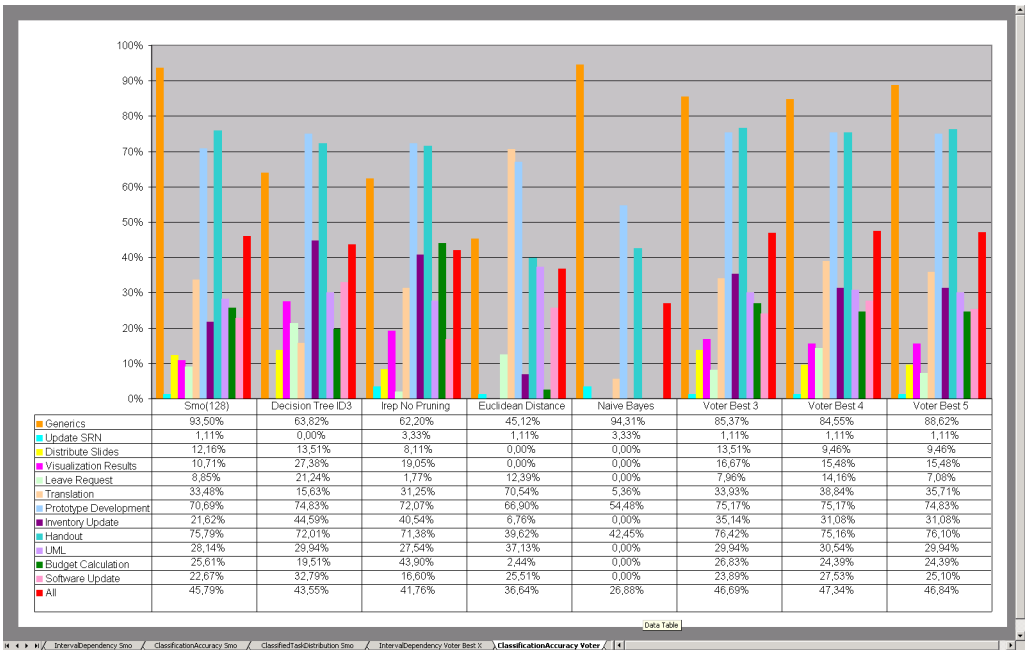
Figure A.22: Transactional Recommender - SMO Classifier - Classification Accuracy

Figure A.23: Transactional Recommender - Voter Classifier - Influence of Interval Seconds



	Smo(128)	Decision Tree ID3	Irep No Pruning	Euclidean Distance	Naive Bayes	Voter Best 3	Voter Best 4	Voter Best 5	Task Distribution
Generics	55,45%	32,06%	22,10%	14,53%	77,30%	47,83%	46,54%	50,07%	12,24%
Update SRN	0,25%	1,44%	0,95%	0,25%	1,05%	0,30%	0,30%	0,30%	4,48%
Distribute Slides	0,90%	1,59%	0,80%	0,30%	0,15%	1,14%	0,85%	0,85%	3,68%
Visualization Results	1,10%	2,84%	1,84%	0,40%	0,15%	1,49%	1,19%	1,14%	4,18%
Leave Request	0,85%	3,19%	0,20%	1,79%	0,05%	0,60%	1,54%	0,45%	5,62%
Translation	5,03%	2,74%	8,56%	40,72%	1,49%	5,33%	7,07%	6,12%	11,15%
Prototype Development	11,95%	16,82%	12,74%	12,94%	10,85%	13,19%	13,69%	13,69%	14,44%
Inventory Update	1,59%	10,95%	11,95%	3,09%	0,00%	5,48%	3,93%	3,73%	3,68%
Handout	13,59%	16,82%	15,48%	6,62%	8,96%	15,08%	14,53%	14,68%	15,83%
UML	3,63%	3,83%	6,82%	12,84%	0,00%	3,98%	4,43%	3,83%	8,31%
Budget Calculation	2,89%	3,04%	16,53%	3,38%	0,00%	2,64%	2,54%	2,04%	4,08%
Software Update	2,79%	4,68%	2,04%	3,14%	0,00%	2,94%	3,38%	3,09%	12,29%

Figure A.24: Transactional Recommender - Voter Classifier - Classified Task Distribution



	Smo(128)	Decision Tree ID3	Irep No Pruning	Euclidean Distance	Naive Bayes	Voter Best 3	Voter Best 4	Voter Best 5
Generics	93,50%	63,82%	62,20%	45,12%	94,31%	85,37%	84,55%	88,62%
Update SRN	1,11%	0,00%	3,33%	1,11%	3,33%	1,11%	1,11%	1,11%
Distribute Slides	12,16%	13,51%	8,11%	0,00%	0,00%	13,51%	9,46%	9,46%
Visualization Results	10,71%	27,38%	19,05%	0,00%	0,00%	16,67%	15,48%	15,48%
Leave Request	8,85%	21,24%	1,77%	12,39%	0,00%	7,96%	14,16%	7,08%
Translation	33,48%	15,63%	31,25%	70,54%	5,36%	33,93%	38,84%	35,71%
Prototype Development	70,69%	74,83%	72,07%	66,90%	54,48%	75,17%	75,17%	74,83%
Inventory Update	21,62%	44,59%	40,54%	6,76%	0,00%	35,14%	31,08%	31,08%
Handout	75,79%	72,01%	71,38%	39,62%	42,45%	76,42%	75,16%	76,10%
UML	28,14%	29,94%	27,54%	37,13%	0,00%	29,94%	30,54%	29,94%
Budget Calculation	25,61%	19,51%	43,90%	2,44%	0,00%	26,83%	24,39%	24,39%
Software Update	22,67%	32,79%	16,60%	25,51%	0,00%	23,89%	27,53%	25,10%
All	45,79%	43,55%	41,76%	36,64%	26,88%	46,69%	47,34%	46,84%

Figure A.25: Transactional Recommender - Voter Classifier - Classification Accuracy

Appendix B

Declaration of Honor

I hereby confirm that I created this master thesis without help of third persons and solely with the literature sources and other resources marked as citations. All passages taken from the literature have been indicated as such. This thesis has not been presented to any other examination commission.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kentlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, September 2008 Matthäus Martynus